ADVANCED COMPUTER PROGRAMMING

A Case Study of a Classroom Assembly Program

F. J. Corbató  J. W. Poduska  J. H. Saltzer

The M.I.T. Press
Massachusetts Institute of Technology
Cambridge, Massachusetts, 1963
PREFACE

The present book is a case study of an assembler/compiler program. It is intended to be an advanced programming text for college students, system programmer trainees, and anyone trying to acquire a general understanding of system programming techniques. We feel that laboratory exercise is an important vehicle for teaching the techniques discussed in this volume. Therefore, the translator program example used must be written in an existing language of an existing computer. We consequently have chosen the FAP language of the IBM 7090 computer to describe the translator program. Other reasons for this particular choice are given in Chapter 1. Any loss of generality is partially offset by the fact that the 7090 is currently the most widely used large-scale computer in the world and one to which many colleges and universities have access.

The motivation for the present work began with the large gap between the usual beginning digital computer programming course and the sophisticated system programming techniques of interest in programming research and development. It was felt that too many students were uncritically using the existing programming systems and were overawed by the apparent complexities in such programs as the original FORTRAN compiler.

In order to serve as an introduction to system programming and to convince the student that the principles of translators are relatively few and basically simple, a Classroom Assembly Program named CAP was written. It was first used in November 1960, in the M.I.T. course 6.251, Digital Computer Programming Systems. Since then, an execution monitor program has been added for the convenience of both students and instructors.

Course 6.251, where CAP has been used, is a one-semester introductory course of 12 units (3 contact hours per week, 9 hours preparation.) The course begins with study of an algebraic language such as FORTRAN or MAD. The next section covers a machine language such as FAP. The third section is devoted to the study of the CAP assembler/compiler. During the semester, the course attempts to present most important contemporary ideas about computer programming. Many of these ideas are then illustrated in the CAP exercise.

Specifically, CAP has been used as follows: Students after studying the translator have been expected to make specified improvements and changes to it, using 6 to 8 computer runs for debugging purposes. (More ambitiously, the students could have written CAP from the specifications, but insufficient computer access prevented this for even the better students.)

For each of the eight semesters that CAP has been taught, the student enrollment, which has been gradually increasing, has been a cross section of the more than twenty departments at M.I.T. Thus we conclude that the average student is able to grasp and enjoy the basic principles of a translator program when it is appropriately presented.

The reader is assumed to be able to program in the FAP machine language sufficiently well to know how to look up features of the FAP assembler or of the 7090 computer in the
IBM published reference manuals. He is assumed also to be acquainted with the Binary Symbolic Subroutine (BSS) linkage and relocation used in the IBM FORTRAN Monitor System (described in the FAP Reference Manual). The book is organized into two major divisions, the description of CAP (five chapters) and the appendices containing listings of the CAP assembler. The compiler part of the program is considered to be advanced material, and the text advises the beginning reader which parts may be safely skipped over.

The appendices include listings of both the assembler-compiler program and of the execution monitor program. The listing of the assembler-compiler is essential to an understanding of the text. The execution monitor listing, while not so important, is included for two reasons. First, an advanced student may make the execution monitor a further case study in advanced programming techniques. Second, it is included for completeness, for the instructor who may wish to adapt it to his needs. It should be noted that the execution monitor program does make use of a few specific features of the current M.I.T. FORTRAN Monitor System and 7090 computer.

Acknowledgment should be given to the efforts of the many teaching assistants who have labored to make the use of CAP effective. Particular mention is made of Neil Haller for his work on the early stages of CAP and introducing the first version of the execution monitor program, and of Neil Barta for his preliminary description of the UPDATE feature of FAP, from which a major part of Chapter 5 is adapted. We also are especially appreciative of the useful comments on the present manuscript made by Neil Barta and Thomas Hastings.

The programs described in this book were developed at the M.I.T. Computation Center, Cambridge, Massachusetts.

Cambridge, Mass. May, 1963

F. J. Corbató
J. W. Poduska
J. H. Saltzer


## CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. CAP USER'S REFERENCE MANUAL</td>
<td>3</td>
</tr>
<tr>
<td>2.1 The CAP Language</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Card Format</td>
<td>3</td>
</tr>
<tr>
<td>Symbolic Location Field</td>
<td>3</td>
</tr>
<tr>
<td>Operation Field</td>
<td>4</td>
</tr>
<tr>
<td>Variable Field</td>
<td>4</td>
</tr>
<tr>
<td>Sequence Number Field</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Pseudo-Operations</td>
<td>4</td>
</tr>
<tr>
<td>2.4 Use of CAP</td>
<td>5</td>
</tr>
<tr>
<td>2.5 Output of CAP</td>
<td>5</td>
</tr>
<tr>
<td>2.6 Restrictions and Error Indications</td>
<td>6</td>
</tr>
<tr>
<td>3. THE CAP ASSEMBLER</td>
<td>7</td>
</tr>
<tr>
<td>3.1 How Does an Assembler Work?</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Pass One, Symbolic Definitions</td>
<td>8</td>
</tr>
<tr>
<td>3.3 The Collation Tape</td>
<td>11</td>
</tr>
<tr>
<td>3.4 Pass Two, Symbolic Evaluation</td>
<td>11</td>
</tr>
<tr>
<td>3.5 VAREVL, Evaluation of the Symbolic Variable Field</td>
<td>12</td>
</tr>
<tr>
<td>How EVAL is Called</td>
<td>16</td>
</tr>
<tr>
<td>3.6 Subprogram Calling Sequences and Definitions</td>
<td>17</td>
</tr>
<tr>
<td>Primary Subroutines</td>
<td>17</td>
</tr>
<tr>
<td>Input and Output Subroutines</td>
<td>18</td>
</tr>
<tr>
<td>Symbol Table Subroutines</td>
<td>19</td>
</tr>
<tr>
<td>Utility Subroutines</td>
<td>20</td>
</tr>
<tr>
<td>4. THE COMPILER OF CØMP PSEUDO-OPERATIONS</td>
<td>22</td>
</tr>
<tr>
<td>4.1 Why A Compiler?</td>
<td>22</td>
</tr>
<tr>
<td>4.2 What Does a Compiler Do?</td>
<td>22</td>
</tr>
<tr>
<td>4.3 Relation of CØMP to CAP</td>
<td>23</td>
</tr>
<tr>
<td>4.4 Precedence</td>
<td>23</td>
</tr>
<tr>
<td>4.5 The Spread Field; CØMPØP</td>
<td>23</td>
</tr>
<tr>
<td>4.6 Compilation of Individual Instructions</td>
<td>23</td>
</tr>
<tr>
<td>4.7 Compilation of Simple Expressions; EXPR</td>
<td>31</td>
</tr>
<tr>
<td>4.8 Temporary Storage and Subroutine GNSTØ</td>
<td>31</td>
</tr>
<tr>
<td>4.9 The Compilation of Terms; TERM</td>
<td>31</td>
</tr>
<tr>
<td>4.10 Review</td>
<td>33</td>
</tr>
<tr>
<td>4.11 Calling Sequence of Compiler Subroutines</td>
<td>34</td>
</tr>
<tr>
<td>5. CAP AS A LABORATORY EXERCISE</td>
<td>36</td>
</tr>
<tr>
<td>5.1 The CAP Laboratory</td>
<td>36</td>
</tr>
<tr>
<td>Extent of Laboratory Assignment</td>
<td>36</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

In an age of increasing complexity, the reader may reasonably ask why he should want to learn the innermost structure of a digital computer programming system. For the day of the renaissance man is indeed past; the intricacies of present-day knowledge as well as the limitation on time for comprehension, of necessity, allow a person to be a specialist in but a limited number of areas. The answers will vary, but it is inescapable that digital computers have already during their short presence become an immensely important device in modern society. As for the future implications, the only issue of debate is whether or not computers are bringing a second industrial revolution as the steam engine heralded the first. Examples of the penetration of computers into our daily activities abound; to name but a few: banking, payroll processing, production and inventory control, income tax processing, satellite orbit computation and tracking, numerically controlled machine tools, airline reservation systems, and military defense communication networks.

Because digital computers have become important, it is inevitable that the accompanying system programs will grow in importance too. For computers reach a high level of effectiveness only when the programming systems allow the ultimate user of the system to program directly—albeit often unknowingly by that name—and thereby avoid intermediary programmers. The development of these direct usage languages is presently limited by the ease and rapidity that suitable translation programs can be written. These translation programs, are variously named problem oriented language processors, compilers, or assembly programs, depending on the language level at which they meet the user. Today, more and more, a computer is incomplete without an accompanying programming system of considerable sophistication.

Moreover, computer systems are still rapidly evolving in many directions: The detailed circuit technology is still making great strides, the logical design is changing to include minicomputers and multiprocessors, and the programming systems are being enlarged to include larger roles such as the time-shared operation of the computer. It is important in this highly fluid state of affairs that others in addition to the system programming specialists have an understanding of programming systems. What is needed for the optimum use of computers in the future is that responsible individuals within computer-affected organizations understand the problems and general techniques of programming systems to the same extent that the problems and techniques of computer hardware are now understood. For without knowledgeable and critical guidance there will be not only many costly abuses of computers but there will be little vision and few ideas for new computer applications.

To give the reader insight into contemporary programming systems, the following chapters will present a case study of the inner structure of a combination assembler-compiler program. The program is called CAP, an acronym for Classroom Assembly Program, and it contains many of the typical features of present-day translators. The case study technique will prove helpful since there are many interrelated factors to consider and
discuss. As well as acquiring an inner knowledge of a translator, the reader of CAP will acquire three additional benefits, namely:

1. The study of detailed programming techniques.
2. How to read and study a large program.
3. How to organize a large program.

For several reasons the CAP program has been written in the FAP symbolic machine language of the IBM 7090 computer. A machine language representation has been specifically chosen because of its concreteness and lack of ambiguity for the reader. This reason is especially pertinent when one considers that one of the principal objectives of the study of CAP is to remove the mystery of system programming and to establish a feet-on-the-ground attitude in the reader. Finally, the FAP language, rather than S0S, for example, has been used in order to have its powerful subprogram feature which allows separate translation and rigid independence of program segments—a feature which greatly assists the initial understanding of a large program.

CAP is weaker than the usual translators, such as FAP, in that it has only subsets and examples of various special features and does not have the machinery for separately translatable subprograms. CAP differs from FAP in style, too, in that it is more elegantly written (that is, in terms of simplicity, brevity, and clarity) and highly organized with many subprograms. The CAP style is in contrast to that of many translator programs in active use where extreme short-cuts have been used in the interest of minimizing operating speed. (Often the short-cuts used are analogous to those for reducing the cost of commercial television receiver and frequently shortsighted from a maintenance point of view.) The basic techniques used in translators remain the same, however, so that CAP is a valid program from which to learn. One feature of CAP that merits comment is that although intermediate tapes are simulated, the program fits entirely in core memory and is independent of intermediate storage devices. Present-day translation programs have frequently overlooked the speed advantages of remaining entirely in core memory particularly while translating short subprograms which should be the major use when a translator allowing subprograms is utilized.

Finally, before proceeding with the remaining chapters, discussion is in order on how to study CAP. Past experience with many students indicates that the following advice is useful:

1. Obtain an understanding of what CAP does from the point of view of a user.
2. Determine the specifications of CAP as a program.
3. Determine the specifications of subroutines PASS1, and PASS2.
4. Starting in PASS1, study the specifications of the successive programs in the hierarchy of subprogram usage. (Omit the compiler.)
5. Starting at the top of the hierarchy, study how each subprogram meets its specifications. Review steps 2 to 4 sufficiently often that you are always sure of what a program is supposed to do before considering how it does it.
6. Remember that all subprograms can only communicate by means of their calling sequences because they are separately translated.
7. When studying, it is a great advantage to know that a program has been debugged. Nevertheless, there will always be sections of program which appear not to work correctly. After spending a reasonable amount of time, if no progress is made, avoid getting bogged down by jotting down on a pad the uncertain point for later discussion with others.
8. The compiler can be studied easily after the basic CAP is understood.
9. The advanced student can improve his program analysis abilities, by studying the execution monitor program, although it is given largely for reference purposes.
Chapter 2

CAP USER'S REFERENCE MANUAL

2.1 The CAP Language

Before we begin to study how the CAP assembly program works, we should pause to determine exactly what job it is intended to do. We can perhaps get the best picture of this job if we examine the user's reference manual for the CAP language. This reference manual is the subject of the present chapter. The brevity of the reference manual is at once an indication of the simplicity of the CAP language and of the assembly program itself.

2.2 Card Format

CAP instructions are typed one to a card as shown in Figure 2.1. Columns 1 to 6 are known as the symbolic location field and may contain a symbol or blanks. Columns 7 and 12 are always blank, leaving room for a three or four letter operation code in the operation field, columns 8 to 11. The variable field begins in column 13 and terminates at the first blank column, or column 73. An arbitrary comment may follow this first blank column. This comment will be ignored by the assembly program as will the sequence number field, columns 73 to 80.

Symbolic Location Field

This field may contain a symbol, a string of one to six characters, at least one of which is nonnumeric, and none of which are the following eleven special characters:

\[ + - * / , = . ' ( ) $ \]
A symbol may be defined only by its appearance in the symbolic location field of some instruction card.

**Operation Field**

This field may contain a mnemonic associated either with one of thirty-four 7090 instructions or one of five pseudo-operations. The allowed 7090 instruction mnemonics are

<table>
<thead>
<tr>
<th>ACL</th>
<th>ANA</th>
<th>CAL</th>
<th>CHS</th>
<th>CLA</th>
<th>CLS</th>
<th>CØM</th>
<th>FAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDP</td>
<td>FMP</td>
<td>FSB</td>
<td>LAC</td>
<td>LAS</td>
<td>LBT</td>
<td>LDQ</td>
<td>LGL</td>
</tr>
<tr>
<td>LGR</td>
<td>LXA</td>
<td>ØRA</td>
<td>PBT</td>
<td>RQL</td>
<td>SLW</td>
<td>STØ</td>
<td>STQ</td>
</tr>
<tr>
<td>SXA</td>
<td>TIX</td>
<td>TMI</td>
<td>TPL</td>
<td>TQP</td>
<td>TRA</td>
<td>TSX</td>
<td>TZE</td>
</tr>
<tr>
<td>XCA</td>
<td>XCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The instructions LAC, LXA, SXA, and TSX are assembled with a tag of 4. The instruction TIX is assembled with a tag of 4 and a decrement of 1.

The allowed pseudo-operation mnemonics are

```
REM   INT   ØCTL   CØMP   END
```

The effect of these pseudo-operations is explained in a later section.

**Variable Field (Operations)**

The variable field specifies the address of an operation. It may contain an expression consisting of a string of symbols and decimal integers connected by the break and grouping characters:

```
+   -   *   (   )
```

All multiplications must be made explicit by the use of the asterisk even if one of the operands is a parenthetical expression. The variable field is evaluated in signed 35 bit integer arithmetic. If the result is negative, it is two's complemented before the final step in which the answer is taken modulo $2^{15}$. The result is combined with the specified operation code by a logical "OR".

**Sequence Number Field**

Columns 73 to 80 may be used for labeling and sequence numbering and are ignored by the CAP assembly program.

### 2.3 Pseudo-Operations

**REM** The REM pseudo-operation is used to introduce an arbitrary remark into the assembly listing. Card columns 1 to 80 will be printed and the card will be otherwise ignored by the assembler. If a symbol appears in columns 1 to 6, it will be ignored.

**INT** INT is a data-generating pseudo-operation. The variable field of the INT pseudo-op consists of signed decimal integers separated by commas and terminating at the first blank column. For each decimal integer, a word is assembled with the decimal integer
inserted in the left half of the word. A comma with no integer following it will cause a word of all zeros to be assembled. A decimal integer may be preceded by a minus sign and must be of absolute value less than $2^{17}$. A symbol, if any, appearing in the symbolic location field will be defined to be the location of the first integer assembled. Succeeding integers will be placed in succeeding locations in core storage.

ØCTL  The twelve characters in card columns 13 to 24 are taken to be octal digits and are used to form a 12 digit octal word in core storage in the next location to be assigned by the assembler. If the characters appearing in columns 13 to 24 are not octal digits, an incorrect word will be generated and no error indication will be made. A symbol in the symbolic location field will be defined to be the location of the generated word.

CØMP  The CØMP pseudo-op specifies that the entire variable field, columns 13 to 72, is taken to be an arithmetic statement which is to be compiled, in much the same manner as in FORTRAN or MAD. Blanks are ignored and commas may be used to indicate tagging. The arithmetic statement must consist of a symbol followed by an equal sign and followed by an arithmetic expression. This expression may consist of symbols connected by the break and grouping characters:

+    -    *    /    (    )

Numbers in the expression will be taken as symbols referring to memory locations. The indicated arithmetic expression will be compiled in floating point arithmetic, and a list of the instructions compiled will appear on the CAP assembly listing. If a symbol appears in the symbolic location field of the CØMP card, its value will be the location of the first compiled instruction.

END  This pseudo-op marks the physical end of the program and defines the entry point to the program to be the value of the expression in the variable field. If a symbol appears in the symbolic location field, it is given the value of the first location not used by the program.

2.4 Use of CAP

CAP is a package of subroutines which is called by

```
.
.
TSX        $CAP,4
.
.
```

The AC should contain the location in core storage into which the first instruction of the symbolic program is to be assembled. When CAP is finished it will leave in the AC the entry point to the program. The sense register (SI) will be nonzero if any assembly errors were noted by CAP.

2.5 Output of CAP

The CAP assembler has two outputs, a printed assembly listing, and a binary machine program. The listing consists of one or more printed lines for each instruction card in the symbolic input deck. This line contains the 80 columns of the original card, the
12 digit octal word which CAP has assembled as well as the octal location in which the instruction has been placed, and pertinent coded error indications. In the case of CØMP pseudo-ops, the CØMP card will be printed and followed by a list of the instructions generated by the compiler in the format described earlier. The assembly listing is written on an output tape for later printing. The binary machine program is left in core storage beginning at the location specified by the program which called CAP.

2.6 Restrictions and Error Indications

1. No more than 100 symbols may be defined. If this restriction is exceeded, further symbols are ignored and a comment is printed at the beginning of the assembly listing, and SI bit 17 will be turned on.

2. All operation codes must be among those listed earlier in this chapter. If an illegal operation code is encountered, it will be treated as zero, SI bit 34 will be turned on, and the letter 'O' will be printed on the assembly listing next to the offending instruction.

3. All symbols appearing in variable fields and CØMP statements must be defined. If an undefined symbol is encountered, it will be given value zero, SI bit 35 will be turned on, and the letter 'U' will be printed next to the offending instruction.

4. The variable field of an INT pseudo-operation must contain only decimal integers, preceded by plus or minus signs and commas. If an illegal character is encountered, that word will be assembled as zero, SI bit 33 will be turned on, and the letter 'E' will be printed on the assembly listing next to the offending pseudo-op.

5. No more than 200 separate elements and break characters may appear in a CØMP statement. If this restriction is exceeded, the CØMP statement is skipped, and SI bit 14 will be turned on.

6. No more than 125 nested parentheses may appear in an arithmetic expression in a variable field. If this restriction is exceeded, an incorrect value may be computed and SI bit 15 or 16 will be turned on, depending on the nature of the parentheses count.

The following two restrictions occur when CAP is run under the Classroom Execution Monitor described in Chapter 5:

7. No more than 150 cards may appear in the symbolic program.

8. The symbolic program must not assemble into more than 256 binary machine instructions or require more than 300 card images to be written on the collation tape.
Chapter 3

THE CAP ASSEMBLER

3.1. How Does an Assembler Work?

In this chapter we shall examine in detail the workings of CAP and of assembly programs in general. While references to the exact coding of CAP are specific to this assembly program, the general discussion and flow charts are common to most assembly programs for most computers.

The purpose of any assembly program is to translate the symbolic cards describing a machine language program into that machine language program. For convenience, this translation can be considered to consist of two operations: First, the mnemonic codes representing machine operations must be replaced by the binary machine codes representing those same operations, and these binary codes must be assigned locations in core storage. Second, the symbolic variable field of each instruction must be evaluated in terms of the symbols appearing in the symbolic location fields of other instructions, and the resulting address must be inserted in the instruction. Consider the following program, written in the CAP language:

```
CAL BITS  GET COUNT.
SLW WORD   SAVE.
HERE TRA HERE  STOP.
BITS INT 6  BIT COUNT.
WORD INT 0  STORAGE FOR BIT COUNT.
```

In order to translate the first instruction, CAL BITS, we need to know two things. First, what is the binary machine code corresponding to the mnemonic CAL? Second, what is the value of the address part of the instruction, that is, what is the value of the symbol BITS? The first question can be answered by reference to a table of operation mnemonics and machine codes, an essential part of any assembler. The second question, however, requires knowledge of which symbolic card has the label BITS. This knowledge can be gained only by going completely through the symbolic deck once to determine the location value of each symbol.

We see, then, that the assembly program must go through the symbolic cards twice. The first pass through the symbolic cards is required to assign each instruction to a place in core storage and thereby to define the value of the symbol, if any, appearing in its symbolic location field. Then, on the second pass through the cards, it is possible to evaluate the variable field of each instruction on the basis of the symbols defined on the first pass.

We may expect, therefore, that CAP will exhibit a basic structure consisting of two passes through the input symbolic card deck. In fact, since CAP is coded in the form of independent subroutines, we shall find that this two-pass structure is handled by two subroutines, named, conveniently, PAss1 and PAss2. These two subroutines are called by
another single subroutine named CAP. (The reader should note that the name CAP will hereafter be used both for the entire assembly program and for the subroutine which calls PASS1 and PASS2. The meaning of any particular usage should be clear from context.) Let us examine a flow diagram of the subroutine CAP, in Figure 3.1.

The CAP subroutine is called by the sequence

```
CAL   ØRG
TSX   $CAP, 4
```

in the main subprogram. (See listings of MAIN and CAP in Appendix I.) ØRG specifies the location in core storage at which the machine language program assembled by CAP is to start.

Subroutine CAP then gives this information as an argument to subroutines PASS1 and PASS2, which perform the two passes through the symbolic card deck mentioned earlier.

Note that subroutines PASS1 and PASS2 upon encountering errors turn on bits in the sense register (SI); subroutine CAP therefore clears the SI before calling each subroutine, and saves its contents upon return. The main program, upon return from CAP, could determine if the assembly was successful by examining the SI, although it does not do this.

### 3.2 Pass One, Symbolic Definitions

It is stated earlier that the purpose of the first pass is to assign each instruction a place in core storage and thereby define all symbols appearing in location fields of the symbolic program. The procedure involved in doing this is, as might be expected, quite straightforward. First, an instruction location counter (ILC) is set to contain the location where the first instruction is to be assembled, which is the origin of the machine language program being generated by CAP. Then, a card is read. If it is not a pseudo-operation, the symbol, if any, appearing in the symbolic location field is defined, the card is put away in a place at which it can be found by pass two, and the ILC is incremented by one. The process is then repeated for the next card. If a pseudo-operation is encountered, some special processing may have to occur. For example, when the END card is encountered, pass one should terminate rather than continue reading cards. A flow diagram of pass one is shown in Figure 3.2.

If we examine the coding of the loop in subroutine PASS1, we find that it takes very few instructions, primarily because the difficult jobs are relegated to subroutines. For example, the box labeled "Read card" is handled by a subroutine named READ1. The entire operation, of determining whether there is a symbol to define and defining it to be the value of the ILC, is handled by another subroutine SYMSTØ. Similarly subroutine WCT1 handles the problem of saving the card for the second pass. If we believe that these subroutines work as their calling sequences specify, the understanding of pass one is greatly simplified.

In fact, the physically largest section of subroutine PASS1 is devoted to processing the pseudo-operations, even though this processing is perhaps the least important function of pass one. Let us examine what must be done when pseudo-operations are encountered. Perhaps the simplest procedure occurs for the pseudo-operation REM. In this case the loop is re-entered after skipping the operations of symbol definition and increasing the
Start

Set ILC.

Save card for pass two.

Read card.

Pseudo-op?

yes → Process pseudo-op

no

Symbol?

no

yes → Store symbol in symbol table with ILC as value.

Increment ILC.

Figure 3.2. Flow diagram of the first assembly pass.

ILC. The only procedure of interest is saving the REM card for pass two. (See Figure 3.3, a flow diagram including pseudo-op processing.)

In the case of the $\mathcal{O}$CTL pseudo-operation during pass one, the only concern is the number of words of storage required (one in this case) and the definition of any symbol appearing in its symbolic location field. Therefore, it can be handled exactly like the ordinary operation codes, that is, by defining the symbol and increasing the ILC by one.

If an INT pseudo-operation appears, the same considerations apply as before. However, the variable field of the INT may specify that several words be generated. (See INT description in Chapter 2.) The variable field always specifies that at least one word should be generated. If there are to be additional words, for each extra word there will be a comma in the variable field. Therefore, the assembler may learn how many words will be generated simply by counting the number of commas in the variable field and adding one. Remember that the only concern of pass one is counting the number of registers used by the source program and defining symbols. The procedure used when an INT is encountered is, then, to test for and define the symbol in its symbolic location field, and to count the number of commas in its variable field. The subroutine $\mathcal{O}$OMMA performs this last step, and also adds one plus the number of commas to the ILC. The loop is then re-entered for the next card.

The operation of the $\mathcal{O}$MP pseudo-operation will not be explained in detail here except to say that the symbol, if any, in columns 1 to 6 is defined, the card is saved for pass two, and a subroutine $\mathcal{O}$MP0P is called to process the pseudo-operation variable field. $\mathcal{O}$MP0P causes the generation of the instruction sequence required to carry out the computation indicated in the variable field and increases the ILC appropriately. The operation of subroutine $\mathcal{O}$MP0P is not essential to an understanding of pass one or the rest of CAP. A full discussion of the subroutine may be found in Chapter 4.

We come finally to the END pseudo-operation. When this card is encountered, pass one is complete except for certain simple terminal procedures. The subroutine END0P must first be called to finish off the work of the $\mathcal{O}$MP0P subroutine by making space at the end of the program for the temporary storage locations required by all the compiled
Figure 3.3. Flow diagram of subroutine PASS1.
instruction sequences. Then, the symbol, if any, in columns 1 to 6 of the END card is defined and the card saved for pass two. Since pass one is now finished the value of the ILC, which is now equal to the first location not used by the object program being assembled, is placed in the AC, and subroutine PASS1 returns to the program which called it.

3.3 The Collation Tape

It has been mentioned several times earlier that pass one must put the symbolic card images away in a place where pass two will be able to find and process them. While in principle it would be possible for pass two to backspace the input tape (or the operator to reload the card reader with the symbolic program), in practice it is much simpler for pass one to write the card images on a second tape, the collation tape. Pass one then ends by rewinding this collation tape, and pass two can begin again with the first card in the symbolic input program.

It is worthwhile noting, also, that when small symbolic programs (say, less than 150 cards) are being assembled, there is no reason why a collation tape is necessary, as there is enough room in the core storage of a 7090 to hold all the card images at once. A common alternate procedure for larger programs is to collect a buffer of, say, 150 cards, then write the entire buffer on a collation tape at once. While the tape write takes place, the assembly program can be processing more input cards and storing them in a second buffer.

Still another method uses two collation tapes, collating half the input cards on one, then starting a rewind so that when pass two begins there will be no wait for tape positioning. The second half of the program is collated on the second tape, which is rewound at the end of pass one, and which will be properly positioned about halfway through pass two when it is needed.

If no collation tape is used, it is still convenient for pass one to call a subroutine to store the cards; the subroutine simply inserts them into a core memory buffer rather than writing a collation tape. Similarly, pass two uses a complementary subroutine which locates and transmits the core buffer rather than reading back from a collation tape.

3.4 Pass Two, Symbolic Evaluation

When all symbols have been defined by pass one, it is possible to finish the assembly by processing each card image in order, and determining values for its operation code and for its variable field. The purpose of pass two, it will be remembered, is to evaluate the operation code and variable field of each card, to assemble the binary machine word required to represent the instruction, and to print an assembly listing containing the original card and the octal equivalent of the machine word generated. Again, the basic procedure is straightforward, although pass two is a little more complicated than pass one. The ILC is again set to start at the origin specified by the program which called CAP.

The main loop of pass two then operates as follows: First, a card is read from the collation tape. If the card does not refer to a pseudo-operation, the operation code is evaluated by comparing it to entries in the operation table. The numeric code of the machine instruction corresponding to the given mnemonic is obtained from this table. Then, the variable field is evaluated. These two results are combined by a logical "OR" and inserted in core storage at the location specified by the ILC. (An alternate procedure might be to store the instruction in an output buffer for punching.) A line is printed
on the assembly listing containing the card image and the octal equivalent of the word that was inserted in core storage. Finally, the ILC is increased, and the loop repeated for the next card.

The main loop of subroutine PASS2 takes but a few instructions, as most of the difficult jobs are handed down to subroutines to perform. The cards are read from the collation tape by subroutine READ2, and the assembly listing is printed by an internal subroutine PRNT1. The most difficult job, evaluation of the variable field on the basis of the symbols defined in pass one, is handled by subroutine VAREVL.

As in pass one, the physically largest section of coding in pass two is that involved in processes not strictly important for an understanding of how pass two works, that is, processing the pseudo-operations, and printing the assembly listing. The pseudo-operations are handled as special cases as they were in pass one, by performing some simple operations and re-entering the main loop at a strategic point. Let us examine them again, one at a time, to learn how they each fit into pass two.

The REM Pseudo-operation again is the simplest of the pseudo-ops. The REM card is printed on the assembly listing, and the loop re-entered at the point where the next card is read. (See Figure 3.4). A slightly different print subroutine is used, as no octal word was generated for the REM pseudo-op and nothing need be printed in the columns normally used for printing the octal word.

The CÔMP pseudo-operation is handled exactly like the REM pseudo-operation in pass two, since all compilation operations were finished in pass one. (See Chapter 4 for details on the CÔMP pseudo-operation.)

The INT pseudo-operation is taken care of very simply by calling a subroutine INTÔP to evaluate the variable subfields and to insert the results in core storage. The INT card is printed on the assembly listing along with the first machine word generated.

The ÔCTL pseudo-operation is handled on the spot by PASS2 as an example of in-line coding. A BCD-binary conversion is performed, the result inserted in core storage, and the ÔCTL card printed on the assembly listing.

As a last step for each of the above pseudo-operations, the pass two loop is re-entered at an appropriate place. In the case of the END pseudo-operation, however, the loop terminates. The variable field of the END card is evaluated by subroutine VAREVL, and this value is saved (and printed) as the entry point to the assembled machine program. Pass two is now complete. The error flags, if any, are placed in the SI, and PASS2 returns to the program which called it.

A comment on the error flags in subroutine PASS2 is in order at this point. Whenever an undefined symbol is encountered in a variable field by subroutine VAREVL, or an illegal operation code by PASS2, or an INT error by subroutine INTÔP, an appropriate bit in the sense indicator register is turned on. The subroutine used to print out the assembly listing examines the SI and prints any error flags next to the instruction being processed. The SI is then set to zero before the next instruction is processed. In addition, one cell is kept throughout pass two which contains the logical combination ("OR") of all the error bits of individual instructions. It is this last cell that is placed in the SI when pass two is finished.

3.5 VAREVL, Evaluation of the Symbolic Variable Field

We now come to the problem of evaluating the symbolic variable field of each instruction; a problem often considered to be the essence of the assembly process. At first glance, given that the values of the symbols which might appear in a variable field have been defined during pass one, we might think that this evaluation would be quite easy. In fact, if we were asked to carry out such an evaluation we would have no difficulty working out the answer in a short time. However, the algorithm needed for the evaluation is
Figure 3.4. Flow diagram of subroutine PASS2.
surprisingly complicated, because of the existence of an implied order of operations in
the mind of the person writing the expression. Consider, for example the following CAP
symbolic instruction:

X  CAL  ALPHA+4* BETA

where ALPHA and BETA are symbols which appear in the symbolic location fields of
cards elsewhere in the program. In evaluating the expression "ALPHA+4* BETA", the
multiplication must be carried out before the addition operation, or else an answer will
be obtained which is different than the one intended by the writer of the expression. Al-
though this order of precedence is a usual convention in mathematical notation, it must
be systematically observed by the assembler when evaluating the expression.

Let us examine a moderately complicated expression and see what sort of combinations
of symbols may appear. After figuring out what procedure is used in each of these cases,
a general procedure will begin to emerge which can be formalized into an algorithm for
the evaluation procedure.

Let us take, as an example, the symbolic expression

+4* ABC- ALPHA+S* 2

and assume that ABC, ALPHA, and S are defined symbols. We first observe that a symbolic
expression can be characterized as a string of elements (symbols or decimal integers)
separated by break characters and terminated by a blank column. The allowed break char-
acters represent the binary operations of addition (+), subtraction (-), and multiplication
(*), and the unary plus and minus sign. For the moment, the ability to handle parentheti-
cal expressions will be ignored. The unary plus at the beginning of the expression, if not
provided by the programmer, is automatically inserted as a first step of evaluation.

To formalize the scan of this expression, let us create three windows which can be
moved across the expression in such a way that the center window always shows us an
element, and the left and right windows show us the break characters on the corre-
spending left and right sides of that element. For example, if the windows were placed on the
above expression as far to the left as possible, we would obtain:

+ 4 5  ABC- ALPHA+S* 2

What does this combination of operands imply? First, the plus sign on the left signals
that we are starting to evaluate a term. The asterisk on the right signals that there are
more things to come in this term, so the saving of the element in the center for a future
multiplication is all we can do. The element is saved in a location named "term" ready
for reference later.

Now, move the windows to the right until the next element falls in the center. We obtain

+4 5  ABC  ALPHA+S* 2

Again examining the left and right break characters to decide what should be done, we
argue as follows: The asterisk on the left tells us to multiply the old value of the term
by the value of the present element. This result may be returned to the storage location
"term". The minus sign on the right signals that the term has come to an end, and that
the value stored away in "term" should be added into the "sum" register for this expression.

Now, move the window to the right again. This time, we obtain

+4* ABC  ALPHA+S* 2

The left window exhibits a minus sign signaling the start of a new term, a negative one
at that. Therefore, we may store away the negative of the value of the present element
in the location "term". The plus sign on the right again signals the end of the term, and
that the value of the term should be added to the "sum" register.
Moving the window once more, we obtain

\[ +4 \text{ ABC-ALPHA} \quad + \quad S \quad * \quad 2 \]

This combination of operators is identical to that found at the beginning of the expression so that we may follow the same procedure. First, on the basis of the plus sign we store away the value of the present element since we are starting a new term. Second, since the \* indicates that there is more to come in this term, we must wait until later elements are brought into consideration.

Finally, with the window in its next and last position, we have

\[ +4 \text{ ABC-ALPHA} + S \quad * \quad 2 \quad 0 \]

This time the situation is similar to one encountered before, except for the lack of an operator in the right window. The left break character again requires us to multiply the value of the term collected so far by the value of the present element. The blank appearing in the right window tells us to add the term into the "sum" register and stop, as the evaluation of the symbolic expression is complete.

Although this procedure seems complicated, let us see if we can develop a flow diagram describing the algorithm. The procedure has the following characteristics: After moving the window, we first examine the break character in the left window, do something about it, then examine the break character on the right. After processing on the basis of this right break character, we move the window and repeat the same series of steps. This procedure is formalized in the flow diagram in Figure 3.5. If we follow the

![Figure 3.5. Flow diagram of subroutine EVAL.](image-url)
flow diagram through for the expression examined previously, we see that it carries out each of the operations described. This flow diagram describes the operation of the subroutine EVAL, which is internal to the subprogram VAREVL. An important procedure which is implicit in this flow diagram is that of evaluating the item appearing in the center window. If the element is a decimal integer, a decimal-to-binary conversion must be made. On the other hand, if the element is a symbol, its value must be looked up. This lookup procedure is done by the subroutine SYMGET which acts as a complement to the subroutine SYMSTØ used during pass one.

How EVAL is Called

EVAL is an internal subroutine of the subprogram VAREVL. The subprogram VAREVL itself simply sets up EVAL and calls it properly; when EVAL has finished evaluating the expression, VAREVL handles the operation of reducing the answer to a core memory location. (See Figure 3.6, a flow diagram of VAREVL.)

Making EVAL an internal subroutine of VAREVL allows EVAL to be defined recursively. That is, if the occasion should arise that EVAL needs to have a subexpression evaluated, it can call on subroutine EVAL to do the job. One might expect to get into difficulty with this procedure, since when EVAL is called recursively, it will change many registers and temporary results. We will see that this difficulty is circumscribed by picking out critical temporary results and saving them in a special way.

In terms of the picture described above, a parenthetical expression may be considered to be an element which appears in the center window. Whenever the center window is determined to contain a parenthetical expression as an element, the element is evaluated by calling the subroutine most able to handle the evaluation of an expression, namely subroutine EVAL. In order to call EVAL, it is necessary to save away temporary results, such as the values of the "term" and "sum" registers that have been collected so that those registers may be used by EVAL for the subexpression evaluation. Then, when EVAL is finished evaluating the subexpression, the "term" and "sum" registers are restored; the evaluation of the original expression continues, using for the value of the element in the center window the answer obtained by EVAL on the recursive call.

Since the parenthetical expression may itself contain another nested parenthetical expression, EVAL must be very careful how it saves away its temporary results, as a second saving of temporary results might destroy the first set.

To handle this problem, two subroutines named SAVE and UNSAVE are used by EVAL. These two subroutines manipulate a last-in, first-out storage array called a push-down list. Each time subroutine SAVE is called, an item or block of items is stored in the list. When subroutine UNSAVE is called, the last item or block stored in the list is retrieved. Successive calls to UNSAVE retrieve items stored by earlier calls to SAVE.

EVAL, then, saves temporary results in the push-down list before calling itself, and retrieves the results later. If the expression requires repeated recursion, the pushdown list will save and restore the temporary variables in the proper order.
Figure 3.7. Flow diagram of EVAL with recursive capabilities.

Figure 3.7 is a flow diagram of EVAL with the ability to handle parenthetical expressions added. The recursive ability of EVAL is not essential to the understanding of the general expression evaluation procedure; it should be ignored in early study by assuming that no parentheses are encountered.

3.6 Subprogram Calling Sequences and Definitions

In this section, the calling sequences and a thumbnail description of each of the utility subroutines used in CAP are described. For reference, the same information about subroutines CAP, PASS1, PASS2, and VAREVL is reproduced here.

**Primary Subroutines**

**CAP**  CAP is called by

\[
\begin{align*}
\text{CAL} & \quad \varnothing \text{RG} \\
\text{TSX} & \quad $\text{CAP}, 4$
\end{align*}
\]

Subroutine CAP causes the symbolic program written on cards and appearing on the input tape to be assembled in core storage starting at the location specified by the address portion of the accumulator.

**PASS1**  PASS1 is called by

\[
\begin{align*}
\text{CAL} & \quad \varnothing \text{RG} \\
\text{TSX} & \quad $\text{PASS1}, 4$
\end{align*}
\]
Subroutine PASS1 performs the first pass of an assembly program over the symbolic cards on the input tape, writes them on a pseudo-collation tape, and defines symbols; assuming that the symbolic program is to start at the location specified by the address portion of the accumulator. If errors are found they are noted in the SI. PASS1 uses index register one to contain the complement of the ILC.

PASS2  PASS2 is called by

CAL  ØRG
TSX  $PASS2, 4

Subroutine PASS2 performs the second pass of an assembly program by reading the symbolic cards appearing on the collation tape. The program is assembled in core storage starting at the location specified by the address portion of the AC, and an assembly listing is prepared on the output tape. PASS2 uses index register one to contain the complement of the ILC. If errors are found they are noted in the SI.

VAREVL  subroutine VAREVL is called by

TSX  $VAREVL, 4
PZE  BUFF

where BUFF is the location of a 14 word buffer containing a symbolic card image. VAREVL will evaluate the variable field starting with the first character of BUFF+2 and continuing to the first blank, comma, or column 73. If any undefined symbols are encountered, SI bit 35 will be turned on.

Input and Output Subroutines

Both PASS1 and PASS2 call several input-output routines to handle tape manipulations. These I/Ø subroutines are

READ1  Read Input Tape, called by

TSX  $READ1, 4
PZE  BUFF

BUFF BSS  14

The 80 columns of a symbolic card are read from the input tape into the fourteen word buffer at BUFF. Note that 80 characters do not quite completely fill the buffer; the last 4 positions may contain arbitrary characters.

WCT1  Write Collation Tape, called by

TSX  $WCT1, 4
PZE  BUFF

BUFF BSS  14
The fourteen word BCI buffer is written on the intermediate tape.

**REWIND** Rewind Collation Tape, called by

```
... TSX $REWIND, 4
...
```

The intermediate tape is marked with an end of file and rewound.

**READ2** Read collation tape, called by

```
... TSX $READ2, 4
    PZE BUFF
    .
    .
    BUFF BSS 14
...
```

Fourteen words of the intermediate tape are read into the buffer at BUFF. READ2 checks that the collation tape has been rewound.

**PRINT** Write on output tape for off-line printing, called by

```
... TSX $PRINT, 4
    PZE A, 0, n
...
```

The n word line image starting in location A is written on the output tape (tape A3). The first character of A (normally blank) is used for carriage control. PRINT counts the lines of output and stops after 300.

**Symbol Table Subroutines**

For forming and searching a symbol table a subroutine package with entries SYMSTØ and SYMGET is used.

**SYMSTØ** The sequence

```
... TSX $SYMSTØ, 4
...
```

will cause the BCD characters in the AC to be scanned (blanks removed), right justified, and inserted in a symbol table together with its value, the complement of IR1. If the symbol is blank, it is ignored and no entry is made in the table.

**SYMGET** The sequence

```
... TSX $SYMGET, 4
...
```
will cause the value of the symbol in the AC (assumed to be scanned and right justified) to be looked up in the symbol table. If the symbol is defined, the value is returned in the AC. If undefined, zero is returned in the AC and SI bit 35 is set on.

Utility Subroutines

CAP also uses a package of utility programs which includes SCAN, CØMMA, SAVE, and UNSAVE.

SCAN SCAN is called by

    TSX $SCAN,4
    ..

on return, the BCD word in the AC is compressed to the right, with blanks removed and leading positions filled with zeros.

CØMMA Subroutine CØMMA is called by

    ..
    TSX $CØMMA,4
    PZE BUFF
    ..

CØMMA counts the number of commas plus one starting with the first character in BUFF+2 and ending with the first blank or column 73. The count is subtracted from index register one. SAVE and UNSAVE manipulate items in a pushdown list.

SAVE SAVE is called by

    ..
    TSX $SAVE,4
    PZE A,0,n
    ..

the n words in registers, A, A + 1, ..., A + n - 1 are placed at the top of the pushdown list and the other items in the list are pushed down n places. (Note that the pushdown effect is achieved by pointers, not by actually moving all the previous entries in the list down in core memory.)

UNSAVE UNSAVE is called by

    ..
    TSX $UNSAVE,4
    PZE A,0,n
    ..

The top n items in the pushdown list are read into locations A, A + 1, ..., A + n - 1 and the other items in the list are pushed up n places.

The pushdown list has a maximum depth of 500 locations. Any attempt to exceed this depth is ignored and SI bit 15 is set. Attempts to retrieve more items than have been stored are ignored and SI bit 16 is set.
Subprogram Calling Sequences and Definitions

Subroutine INTØP is used to evaluate variable fields of the INT pseudo-op during pass two.

\[\text{INTØP}\] INTØP is called by

\[\begin{align*}
\text{TSX} & \quad \$\text{INTØP, 4} \\
\text{PZE} & \quad \text{BUFF}
\end{align*}\]

where BUFF+2 is the address of the first location of the buffer containing the variable field. INTØP scans the variable field and converts each decimal subfield (as delineated by commas) to a binary number; shifts the number obtained into the decrement; and stores it in the next location in the program being assembled, assuming that index register one contains the complement of the ILC. INTØP then increments the ILC and repeats the operation for the next subfield.

Subroutine ENDØP is used at the end of pass one to reserve temporary storage for COMP pseudo-ops.

\[\text{ENDØP}\] ENDØP is called by

\[\begin{align*}
\text{TSX} & \quad \$\text{ENDØP, 4}
\end{align*}\]

Control returns to the caller after ENDØP changes the C(IR1) by the proper amount and enters the symbol TEM into the symbol table.

\[\text{OPTBL}\] The first word in $OPTBL$ is a control word containing in its address the location of the first item in the operation table and in its decrement the length of the operation table; the rest of OPTBL consists of pairs of entries, a right-justified BCD mnemonic paired with the binary machine code for that mnemonic.

Subroutine COMPØP and the subroutines it calls are described in Chapter 4.
Chapter 4

THE COMPILER OF CØMP PSEUDO-OPERATIONS

In this chapter we will examine in detail the operation of the set of subprograms which compile arithmetic for CØMP pseudo-operations. The material under discussion is of an advanced nature and not essential to an understanding of the CAP assembly program. A beginning reader may skip this chapter, as the material in the sequel will not make reference to the compiler. The reader is assumed to be familiar with an algebraic language such as FØRTRAN, ALGØL, or MAD.

4.1 Why a Compiler?

Compilers exist to free the programmer from worry about coding details while working with algebraic calculations. The compiler can take care of the coding details, and the programmer need only concentrate on setting up the proper equations.

The primary reason for including a compiler in CAP is educational. We shall see the close similarity between the internal processes of assemblers and compilers; some of the mystery as to how compilers work will thereby disappear.

Another reason for including a compiler is to provide a contrast with the macro-operation processors found in many present-day assembly programs. A compiler is an often overlooked alternative and provides a flexibility of expression which the macro-processor cannot obtain.

4.2 What Does a Compiler Do?

The point of the compiler is very simple. If the programmer writes on a card a statement

$$\text{CØMP} \quad Y = \text{ALPHA} + \text{BETA}$$

the program which results is identical to that which would have resulted if the programmer had instead given the instructions

$$\begin{align*}
\text{CLA} & \quad \text{ALPHA} \\
\text{FAD} & \quad \text{BETA} \\
\text{STØ} & \quad Y
\end{align*}$$

We see, then, that the purpose of the compiler is to generate a program to perform the algebraic computation indicated by the symbols and break characters in the variable field of the CØMP statement.
There are several algorithms available to perform the compilation. In the CAP compiler, a nonrecursive procedure contrasts with the recursive procedure used for evaluating expressions in subroutine VAREVL, discussed in Chapter 3. We will see that the algorithm is a collection of simple, straightforward ideas combined in such a way as to produce a sophisticated result.

4.3 Relation of COMP to CAP

We recall that when the CAP assembler encounters a COMP pseudo-operation during pass one, it calls a subroutine named COMP.

COMP and the collection of subroutines which it calls compile the symbolic machine instructions in the CAP language required to carry out the computation called for by the COMP statement. The compiler writes these symbolic instructions on the collation tape in the same format as CAP language symbolic instructions which the programmer writes and the order in which they are to be performed. The compiler increases the ILC by the number of instructions compiled, and returns control to subroutine PASS1 to continue the first assembly pass. By writing symbolic cards on the collation tape during pass one, the compiler thereby discharges its responsibility; the symbolic instructions on the collation tape will be assembled by the second assembly pass as would instructions provided by the programmer himself.

4.4 Precedence

The language available to the COMP programmer allows the use of addition, subtraction, multiplication, and division—with parentheses as grouping characters. Since the programmer will wish to attach an order of precedence to these operations, the compiler must take that order into account when creating the symbolic program. The order of precedence used is the following:

parenthetical expressions
multiplication and division
addition and subtraction

This precedence table corresponds to the table commonly assumed by mathematicians. It states, for example, that in the expression

A + B/C

the division is to be carried out before the addition.

4.5 The Spread Field: COMP

The subroutine called to compile COMP pseudo-operations is COMP. COMP operates in two passes. In the first pass, it scans the variable field of the COMP card, ignoring blanks, and separates the symbols and break characters one to a word in a buffer known as the spread field. For example, if the variable field contains

SUM = G1 + G2 + G3/SIX
pass one of CôMPØP would produce a spread field containing in successive locations

\[
\text{SUM} = G1 + G2 + G3 \nonumber \\
/ \nonumber \\
\text{SIX} \nonumber 
\]

Later scans may now search the spread field for break characters with a simple search loop. Symbols which are longer than six characters are permissible. They will be broken up and stored in successive words in the spread field. Since the comma is not a break character, the sequence of characters ABC,1 will be considered to be a single symbol and stored appropriately. When compiled as the address of an instruction, this symbol could represent a tagged address.

All scans of the spread field will ignore a zero appearing within the spread field. The value of this property will become clear later when we see how the spread field is modified as the expression is compiled. An alternative procedure with similar flexibility is to place successive items of the spread field in a string pointer list.

Having re-expressed the arithmetic statement to be compiled in a form easier to work with, subroutine CôMPØP proceeds with the actual compilation. A scan is made for a parenthetical expression which is in some sense "innermost." That is, it is to contain no parenthetical expressions. The procedure for finding such an "innermost" expression is as follows: Scan the spread field starting at the top for left and right parentheses, leaving markers behind at the left parentheses, and stopping at the first right parenthesis. The last left parenthesis marker and the position of the right parenthesis define an "innermost" parenthetical expression. A subroutine named EXPR is now called, with arguments consisting of the pointers to the left and right ends of the parenthetical expression, and the location of the beginning of the spread field. Subroutine EXPR will compile the symbolic CAP language program necessary to compute the expression within the parentheses and write this symbolic program on the collation tape. EXPR will then modify the spread field by replacing the left parenthesis, the entire expression within the parentheses, and the right parenthesis with zeros. The last instruction in the symbolic CAP language program generated by EXPR will be an instruction to store the result of the computation in a temporary storage location. The symbolic name of this temporary storage location is inserted directly in the spread field by EXPR in one of the locations formerly occupied by the parenthetical expression. The symbol TEM+nn will always fit into the space vacated by the original expression. This is one of the reasons for choosing to spread out the original expression into a spread field.

At this point, the "innermost" parenthetical expression is compiled. CôMPØP now starts over again, looking for a new "innermost" parenthetical expression in the modified spread field. Since the old expression, along with its parentheses, was replaced by a single symbol in the spread field, CôMPØP can scan for a new "innermost" parenthetical expression exactly as it did before. It is now clear why zero words are ignored within the spread field. Whenever the compiler writes instructions on the collation tape, it replaces the symbols and operators within the spread field leading to the compilation
of these instructions by zeros. Later scans of the spread field ignore the presence of the zero positions, as nothing more is to be compiled from the information that was once contained there.

CØMPØP iterates in the manner described; first locating an innermost parenthetical expression, and then calling upon EXPR to compile the expression. EXPR removes the expression from further consideration by modifying the spread field.

Eventually, CØMPØP will reach a situation in which the spread field contains no parenthetical expressions. Instead, it will contain a simple expression preceded by a symbol and an equal sign. In this case, subroutine EXPR is again called with parameters indicating the beginning and end of the simple expression and with an additional parameter specifying that the program compiled is to leave its result in the AC rather than in temporary storage. EXPR again generates symbolic instructions, writes them on the collation tape, and modifies the spread field by replacing all elements compiled by zeros. Upon returning to CØMPØP the compilation is nearly completed except for storage of the final result. Subroutine CØMPØP then generates the necessary STØ instruction to complete the compilation. Let us follow this procedure through for a moderately complicated expression. Consider the following CØMP pseudo-operation

\[
CØMP \quad Y = ((A+B)*(E-C*DL)+END)*F+L1
\]

Figure 4.1 shows the spread field and instructions compiled in succeeding steps. Figure 4.2 is a flow diagram of CØMPØP.

Step 1. CØMPØP places the variable field in the spread field (Figure 4.1a) and scans for left and right parentheses, starting at the top, ending with the first right parenthesis. (See Figure 4.1b.) It then calls EXPR to compile this "innermost" expression. EXPR will write the instructions indicated as "step one" in Figure 4.1f, on the collation tape and modify the spread field to that shown in Figure 4.1c.

Step 2. CØMPØP scans again for left and right parentheses and calls EXPR to compile the expression found. EXPR writes on the collation tape the instructions indicated as "step two" in Figure 4.1f, and modifies the spread field to that shown in Figure 4.1d.

Step 3. One more scan for parenthetical expressions results in a call to EXPR and compilation of instructions indicated as "step three" in Figure 4.1f. EXPR modifies the spread field to appear as in Figure 4.1e.

Step 4. The scan for parentheses fails this time. CØMPØP calls EXPR to compile the remaining simple expression and specifies that the result of the computation be left in the AC. EXPR compiles the instructions labeled "step four."

Step 5. CØMPØP compiles an STØ instruction with a symbolic address consisting of that variable to the left of the equal sign. The compilation is now complete.

CØMPØP keeps track of parenthetical expressions by means of pointers to positions in the spread field. An alternative procedure is to push successive field items down in a push-down list searching for a right parenthesis. Then, the subroutine compiling the expression can retrieve items back to the last left parenthesis.

Note that we have not yet learned how EXPR compiles the symbolic arithmetic instructions and places them on the collation tape. We are analyzing the compiler from the "outside in" and are still at a stage where the organization of the compiler is the most important thing to be learned. Having established the procedure by which parentheses are handled, we are now ready to begin studying the details of instruction creation.
The Compiler of CØMP Pseudo-Operations

Figure 4.1. Successive spread fields and resulting compilation for CØMP $Y = ((A+B)*(E-C*DL)+END)*F+L1$. 

(a) (b) (c) (d) (e) (f)
Figure 4.2. Flow diagram of subroutine COMPØP.
4.6 Compilation of Individual Instructions

In the fifth step in the example above, subroutine CØMPØP had to compile the instruction STØ Y. To write this instruction on the collation tape, a package of subroutines is used which manipulate a collation tape buffer and write on the collation tape. The collation tape buffer is a 14-word buffer which is used to collect a symbolic card image.

The first subroutine in this package is PIVAR. (Place in variable field.) Its calling sequence is

\[ \text{TSX } \$\text{PIVAR, 4} \]

PIVAR takes the contents of the AC as a BCD word, and inserts that BCD word in the next available space in the variable field of the collation tape buffer. Columns 13 to 18 are filled in by the first call to PIVAR, columns 19 to 24 on the next, etc.

The last piece of information known about any instruction is always the operation code. Subroutine GENØP inserts the operation code and writes the collation tape buffer on the collation tape. Its calling sequence is

\[ \text{TSX } \$\text{GENØP, 4} \]

\[ \text{BCI } 1, \text{ opr} \]

where "opr" is the operation mnemonic to be inserted in the operation field. GENØP inserts the instruction code into the operation field (columns 7 to 12) writes the entire collation tape buffer on the collation tape, and clears out the buffer with blanks, resetting PIVAR to store in columns 13 to 18. Thus the sequence required to generate the STØ Y instruction in step five, above, is

\[ \text{CAL } \text{FLD, 1} \]

\[ \text{GET SYMBOL FROM SPREAD FIELD.} \]

\[ \text{TSX } \$\text{PIVAR, 4} \]

\[ \text{INSERT IN VAR FIELD.} \]

\[ \text{TSX } \$\text{GENØP, 4} \]

\[ \text{GENERATE STØ ØP.} \]

When it compiles instructions, subroutine EXPR also uses the subroutines PIVAR and GENØP.

4.7 Compilation of Simple Expressions; EXPR

Subroutine EXPR has the responsibility of compiling parentheses-free expressions. This responsibility includes the proper handling of precedence below the level of parenthetical expressions. EXPR handles precedence by making two passes over the symbolic expression; during the first pass, all terms (symbols connected by asterisks and slashes) are compiled leaving the expression in the form of a summation of individual elements (subroutine TERM compiles the terms). In the second pass over the expression, EXPR
compiles the necessary add and subtract instructions to complete the summation. Let us consider a typical spread field expression that EXPR is to compile. The expression comes from Step 2 of the previous example.

\[
\begin{align*}
E &= C - C * DL \\
\end{align*}
\]

In the first pass, EXPR locates terms containing more than one symbol. In the given expression, the second term falls into this category. Therefore, EXPR calls subroutine TERM with parameters pointing to the upper and lower boundaries of the term C*DL. Subroutine TERM compiles a program which computes the value of the term and inserts the answer into temporary storage. In this case the program written on the collation tape is

\[
\begin{align*}
LDQ & C \\
FMP & DL \\
STO & TEM \\
\end{align*}
\]

TERM will also modify the spread field by replacing the elements of the term with zeros, and inserting the name of the temporary storage location into the spread field in an appropriate place. When TERM finishes, the spread field will appear as follows:

\[
\begin{align*}
E &= - \\
TEM &= 0 \\
0 &= 0 \\
\end{align*}
\]

Since there are no more terms in our sample expression, pass one of EXPR is complete, and pass two begins. In pass two, EXPR compiles and writes on the collation tape a program to perform the summation of the elements in the expression.

The second pass consists of the following steps, indicated in the flow diagram in Figure 4.3.

1. Scan the spread field from the top, looking for the end of the first symbol. If an initial minus sign is passed, set a switch.

2. Compile the instruction CLA or CLS (on the basis of the switch set in Step 1) with a symbolic address consisting of the symbol obtained in Step 1, using PIVAR and GENOP. Replace the operator and the symbol in the spread field with a zero.

3. Continue scanning the spread field for the end of the next symbol. Again, if an initial minus sign is passed, set a switch.

4. Compile the instruction FAD or FSB (on the basis of the switch set in Step 3) with a symbolic address consisting of the symbol obtained in Step 3, using PIVAR and GENOP. Replace the symbol and the operator in the spread field with a zero.

5. Repeat Steps 3 and 4 until the end of the expression is reached. Now, if requested, compile an instruction to store the result in a temporary location. The second pass is now complete, and the expression has been compiled.
Figure 4.3. Flow diagram of subroutine EXPR.
4.8 Temporary Storage and Subroutine GNSTØ

The last step in subroutine EXPR was compilation of an instruction to store the AC in a temporary location. What symbolic address should be placed in the STØ instruction, and how can temporary storage be reserved? Subroutine GNSTØ provides this service. The calling sequence

\[
\text{TSX \ $GNSTØ,4}
\]

will bring into the accumulator the symbol TEM+n where n is one less than the number of times GNSTØ has been called. Subroutine GNSTØ will also keep track of the total number of temporary locations used so that subroutine ENDØP can reserve space at the end of assembly pass one. The first call to GNSTØ brings back the symbol TEM; later calls produce symbols such as TEM+1, etc. The instruction

\[
\text{STZ* \ $NSTØ}
\]

resets GNSTØ so that the next call starts again with the symbol TEM. Since separate COMP statements are independent, they can use the same temporary storage locations, and COMPØP resets NSTØ at the beginning of each new COMP statement.

The sequence used by EXPR to compile the store instruction is, then,

\[
\begin{align*}
\text{TSX \ $GNSTØ,4} & \quad \text{GET TEMPØRARY SYMBOL.} \\
\text{SLW \ FLD,1} & \quad \text{INSERT IN SPREAD FIELD.} \\
\text{TSX \ $PIVAR,4} & \quad \text{PLACE IN VARIABLE FIELD.} \\
\text{TSX \ $GENØP,4} & \quad \text{GENERATE STØ ØP.} \\
\text{BCI 1, STØ} & \quad \text{.}
\end{align*}
\]

4.9 The Compilation of Terms; TERM

When EXPR encounters a term consisting of symbols connected by asterisks and slashes, it calls subroutine TERM to compile instructions which compute the value of the term and leave the result in temporary storage. Subroutine TERM performs this compilation by scanning the term in much the same manner as subroutine VAREVL (see Chapter 3) noting for each symbol the break character on its left and on its right. The break character on the left may be the beginning of the term, an asterisk, or a slash. The one on the right may be the end of the term, an asterisk or a slash. Thus a symbol may have one of nine pairs of break characters associated with it. Since the instructions compiled in each of the nine cases is different, a nine-way branch must be made for each symbol. The flow diagram in Figure 4.4 illustrates this nine-way branch. The scan of the term begins at the left (or top, in terms of the spread field).

Let us consider a simple term, and follow the operation of TERM through the flow diagram. Suppose TERM is to compile the following spread field:

\[
\begin{align*}
\text{C} \\
\text{D} \\
\text{E} \\
\text{F}
\end{align*}
\]
Upon scanning for the first symbol, we find that the left break is the beginning of the term, the right break an asterisk. Following the flow diagram, we see that the instruction LDQ C is compiled in preparation for the multiply operation. We may note that in this case, the compilation leaves the result in the proper register so that the next instruction FMP will operate correctly. If the right break character had been a slash, the instruction CLA C would have been compiled instead. We will see that the algorithm leaves the result in the proper register in all cases.
The scan now resumes. The next symbol has an asterisk on the left and an asterisk on the right. The asterisk on the left signals that we should compile the instruction FMP D; the asterisk on the right warns of a coming multiplication, so the result must be returned to the MQ with an XCA instruction.

Resuming the scan once more, we find that the third symbol has on the left an asterisk, on the right a slash. Again, the asterisk on the left signals that the instruction FMP should be compiled; however, the slash on the right indicates that the next operation will be division. Therefore, the result is left in the AC in proper position for the FDP instruction.

Returning to the scan for the fourth and final time, we find the symbol $F$ surrounded by a slash on the left and the end of the term on the right. The slash calls for a division operation, so the instruction FDP $F$ is compiled. The end-of-term break indicates that we are almost finished. A temporary storage location is generated by GNSTØ and the instruction STQ TEM is compiled. Note that if the last operation had been a multiplication, the last instruction would have been STØ TEM instead.

Now, compilation of the term is finished. Although it has not been mentioned before, the spread field was reset to zero during the scan, and, at the end, symbol TEM was placed back into the spread field. The final result of the compilation by TERM is as follows:

<table>
<thead>
<tr>
<th>Spread field</th>
<th>Collation tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEM</td>
<td>LDQ C</td>
</tr>
<tr>
<td>0</td>
<td>FMP D</td>
</tr>
<tr>
<td>0</td>
<td>XCA</td>
</tr>
<tr>
<td>0</td>
<td>FMP E</td>
</tr>
<tr>
<td>0</td>
<td>FDP F</td>
</tr>
<tr>
<td>0</td>
<td>STQ TEM</td>
</tr>
</tbody>
</table>

4.10 Review

With the study of subroutine TERM, we have completed our examination of the compiler. A brief review of the essential points covered may help place those points in the proper perspective.

The compiler operates during the first assembly pass of CAP. The compiler places the instructions generated on the collation tape for processing by the second assembly pass just as though the programmer had provided them.

Subroutine CØMPØP coordinates the compilation. CØMPØP goes over the symbolic expression in two passes. During the first pass, it places the symbolic expression in the spread field — one symbol or break character to a memory location.

In the second pass it evaluates the expression from the innermost set of parentheses outward with the help of subroutine EXPR. Subroutine EXPR also operates in two passes. In the first pass, EXPR reduces the expression to a summation by calling on subroutine TERM to compile the instructions to compute the individual terms. The second pass of EXPR compiles the instructions needed to compute the resulting summation.

During all phases of the compilation, the compiler modifies the spread field as it generates instructions and places them on the collation tape. Subroutines GENØP, PIVAR, GNSTØ, and ERASE help put together symbolic instructions and write them on the collation tape.

When the compilation is finished, control returns to CAP to continue assembly pass one.
4.11 Calling Sequence of Compiler Subroutines

This section describes the calling sequences of each of the subroutines of the compiler and presents for easy reference a thumbnail sketch of the external characteristics of each subroutine.

**COMPØP** Subroutine COMPØP is called by

<table>
<thead>
<tr>
<th>TSX</th>
<th>COMPØP,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PZE</td>
<td>BUFF</td>
</tr>
</tbody>
</table>

where BUFF is the first location of a 14-word buffer containing the symbolic COMP card. COMPØP compiles the instructions necessary to perform the arithmetic specified by the variable field of the card in the buffer, writes these instructions on the collation tape, and increases the value of the ILC (assumed to be stored in complement form in index register one) by the number of instructions compiled.

**EXPR** Subroutine EXPR is called by

<table>
<thead>
<tr>
<th>TSX</th>
<th>EXPR,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PZE</td>
<td>LI, T, RI</td>
</tr>
<tr>
<td>PZE</td>
<td>FLD</td>
</tr>
</tbody>
</table>

where FLD-LI is the address of the left break and FLD-RI is the address of the right break. EXPR takes a string of symbols connected by + - * or / and compiles the result in floating point. If T = 0, the result is placed in temporary storage. Otherwise, the result is in the AC. The spread field is modified accordingly.

**TERM** Subroutine TERM is called by

<table>
<thead>
<tr>
<th>TSX</th>
<th>TERM,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PZE</td>
<td>LI, 0, RI</td>
</tr>
<tr>
<td>PZE</td>
<td>FLD</td>
</tr>
</tbody>
</table>

where FLD-LI is the address of the left break, and FLD-RI is the address of the right break. TERM takes a string of symbols connected by + or / and compiles the result in floating point. The compiled program places its result in temporary storage, and TERM modifies the spread field accordingly.

The following subroutines are used to form symbolic instructions:

**PIVAR** Subroutine PIVAR (place in variable field) is called by

| TSX | PIVAR,4 |

PIVAR takes the C(AC)_{p, 1-25} as a BCD word and stores that word in the next available location in the collation tape buffer. On the first call to PIVAR, the next available location is the first word in the variable field position of the buffer.

**ERASE** Subroutine ERASE is called by

| TSX | ERASE,4 |

Subroutine ERASE clears the collation tape buffer, replacing all words with blanks, and resetting PIVAR so that on the next call it will start at the beginning of the variable field.
GENØP Subroutine GENØP is called by

TSX $GENØP,4
BCI 1, opr

where the letters "opr" are the symbolic operation code desired. GENØP will take the symbolic operation code in location 1,4 and insert it into the operation field of the collation tape buffer. It will then write the buffer on the collation tape and call subroutine ERASE to clear the buffer so that it may be used again.

GNSTØ Subroutine GNSTØ is called by

TSX $GNSTØ,4

Subroutine GNSTØ returns to the caller after placing in the AC p.,1-35 a symbol of the form TEM+n where n is one less than the number of times that GNSTØ has been called. Entry point NSTØ will contain this number; and if NSTØ is reset to zero, n will be reset to zero for the next call to GNSTØ. GNSTØ keeps track of the largest n ever encountered and leaves it in a location where it is accessible to subroutine ENDØP for purposes of assigning temporary storage at the end of the first assembly pass of CAP.
Chapter 5

CAP AS A LABORATORY EXERCISE

CAP finds application both in the classroom and in the laboratory. In the laboratory the student modifies or improves the assembler, for example, by adding pseudo-operations to make the CAP language more flexible or by improving the internal operations of the assembler. Appendix C contains a list of suggested modifications.

This chapter is divided into two parts to correspond, roughly, to material of greater interest to the student and to his instructor, respectively. No clear line can be drawn between these interests, of course, as the instructor will wish to read the entire chapter and an advanced student will find much of interest in the second part.

5.1 The CAP Laboratory

The CAP assembly program was written with expansion in mind. Thus, although there might be simpler ways to perform some of the operations called for in the original CAP language, extension of these operations might be difficult if a simpler, less general, approach had been used in the original coding. There are also several examples throughout CAP of points onto which additional coding may be easily attached. An analogy would be the complicated highway interchange with one blocked exit at a point where a new highway is to be built someday.

The suggested modifications represent changes which are at once useful, educational, and not too difficult, when the operation of the original assembler is well understood.

When CAP is used in the laboratory, the main program which calls CAP is replaced by an execution monitor program to aid in debugging the modifications. This execution monitor provides aid in case the modified assembly program gets into a loop or comes to a stop, and it provides a postmortem when the CAP assembly is finished.

Also, in the laboratory, the input-output subroutines are replaced with an L/α simulator package to speed up testing; this simulator provides as CAP input a symbolic test program for assembly and simulates the collation tape with a core buffer.

Extent of Laboratory Assignment

A typical laboratory assignment might be the following: The instructor selects a set of modifications totaling in value about 200 "points" as required modifications. (See Appendix C for point values.) The student then selects additional modifications worth about 100 points. The student is permitted eight or nine computer "runs" to attempt to get all 300 points of modifications working correctly.

Evaluation of the student's work is done on the basis of a brief written report describing the modifications attempted and the degree of success in achieving modification. Printed computer output should accompany the report as evidence of correct operation of the modified assembly program.
How CAP Is Modified

Two different procedures have been used to allow the modification of CAP. In the first and simpler procedure, the student makes a copy of the symbolic decks of all the subroutines basic to the assembler and, if desired, the compiler. He then makes changes to this deck of 1000 to 2000 cards and submits it for assembly by FAP and testing under the execution monitor.

If this procedure is used, the reader may wish to skip the next sections and proceed immediately with the discussion of testing of the modified assembly program (Section 5.3).

5.2 UPDATE

If a large class uses CAP as a laboratory exercise, the above procedure can lead to the processing of a very large number of cards. An alternate procedure involving the UPDATE feature of FAP can significantly cut down on the number of cards used. Under this procedure, the unmodified CAP subprograms are placed in symbolic form on a single UPDATE input tape for all students, and each student need only submit cards corresponding to the changes he wishes to make in the subprograms. The UPDATE pseudo-operations of the FAP language control the merging of the student's changes with the original symbolic programs and the assembly of the merged programs.

The UPDATE procedure has the disadvantage that the student must learn the UPDATE language in order to modify CAP. However, the advantages of a small input deck are significant both in time saved preparing input tapes for the computer and in added reliability of a smaller deck of cards.

All features of the UPDATE language necessary for the successful modification of CAP will be discussed here. The FAP Reference Manual contains additional information.*

The Use of UPDATE

Images of the cards submitted for a run are written ahead of run time on the System Input Tape by off-line card-to-tape equipment. When programs are assembled normally on the 7090 (without UPDATE), FAP reads the card images from the System Input Tape and processes them one at a time. When UPDATE is used, two more tapes are involved: the UPDATE Input Tape and the UPDATE Output Tape. In CAP, only the UPDATE Input Tape is used.

The UPDATE Input Tape contains the unaltered symbolic versions of the CAP subroutines as shown in the listings in Appendix A. The serialization in columns 73 to 80 on the lists is also on the UPDATE Input Tape and is used by FAP to determine the order of processing card images from the System Input Tape and the UPDATE Input Tape.

Because the UPDATE facility is a part of FAP the first card of any deck submitted using UPDATE must be

* FAP

This card causes control to be transferred to FAP. FAP retains control until an END card is processed. It is important to keep in mind that the program assembled begins at the * FAP card on the System Input Tape and terminates with the first END card processed; this END card may be on either the System Input Tape or the UPDATE Input Tape. Assembly of another subprogram requires another * FAP card.

The use of four UPDATE pseudo-operations (UPDATE, DELETE, DELETE THRU, and SKIPT©) will be described. UPDATE operations are FAP pseudo-operations and, as such, begin in column 8 of the card.

The UPDATE Pseudo-Operation

The UPDATE pseudo-operation specifies the use of the UPDATE feature of FAP. A card with UPDATE punched in the operation field follows the © FAP card. The variable field, beginning in column 16, specifies the details of the UPDATE run. The first subfield contains the logical tape number of the tape unit on which the UPDATE Input Tape has been mounted. In the following examples we will assume that the UPDATE Input Tape is mounted on logical tape drive 11. The other subfields of the UPDATE card specify features not used in CAP and should be left blank. Hence the first two cards in each CAP UPDATE assembly are

© FAP
UPDATE 11

Adding and Replacing Cards

Assembly, initiated by the © FAP and UPDATE cards, continues as card images of FAP instructions are read from the normal System Input Tape and the UPDATE Input Tape one at a time in serial order. A serialized card image on the System Input Tape is assembled before a card of equal or higher serialization but after a card of lower serialization on the UPDATE Input Tape. Whenever FAP encounters card images of equal serialization on the two tapes, the card image on the System Input Tape is assembled in place of the card image on the UPDATE Input Tape. If there is no serialization on the card image on the System Input Tape, the card image is immediately assembled. (See Figure 5.1, a flow diagram of UPDATE.)

More than nine cards can be inserted between two consecutive cards already on the UPDATE Input Tape by giving the first card to be inserted a serial number between the two cards on the UPDATE Input Tape. The remaining cards to be inserted at this point in the subprogram are not serialized.

Changes can be made in increasing order of serialization only.

Deleting Cards from Programs on the UPDATE Input Tape

To remove a card from a program on the UPDATE Input Tape, the DELETE pseudo-operation is used. When FAP reads a card from the System Input Tape that has DELETE in its operation field, cards are assembled from the UPDATE Input Tape until a card image with serialization equal to that of the DELETE card is found. FAP does not assemble this card image from the UPDATE Input Tape; normal updating and assembly continue with the next card from each tape.

If many consecutive cards are to be deleted from programs on the UPDATE Input Tape, the DELETE THRU pseudo-operation may be used. When FAP reads a card that has DELETE in its operation field and the letters THRU in the variable field, no more card images from the UPDATE Input Tape are assembled until a card of serialization higher than that of the DELETE THRU card is found on the UPDATE Input Tape. © FAP will then resume normal updating and assembly.

© As of May, 1962, the M.I.T. version of FAP requires THRU in columns 15 to 18; this differs from the FAP Reference Manual description of DELETE THRU.
To delete a block of cards from the middle of a program: First, insert a DELETE card with serialization of the first card in the block. This DELETE card should be followed by a DELETE THRU with serialization equal to the serial number of the last card to be deleted. DELETE THRU will delete a card of equal but not higher serialization. The input tapes should never be moved backward while updating a program.

The Necessary END Card

To insure proper operation of UPDATE, the last card of the input deck for each subprogram updated must be a serialized END card. The serialization of the END card in the input deck must be identical to that of the END card on the UPDATE Input Tape for the subprogram being updated.

Bypassing Assembly of Subprograms

The UPDATE Input Tape will be rewound before the job starts and we may assume that it is properly positioned to begin assembly of the first subprogram on the tape. The order of subprograms on the UPDATE Input Tape is specified in Figure 5.2. The order is the same as on the CAP listings.

The first

* FAP
UPDATE 11

would therefore, start assembly of subprogram CAP. At the end of this assembly the UPDATE Input Tape would be positioned ready to start assembly of the second subprogram. The next

* FAP
UPDATE 11

would start assembly of PASS1, and so forth.

Most of the suggested alterations to CAP require changes to only a few of the subprograms. Therefore, it would be wasteful of machine time to assemble all of the CAP subprograms during each run. Assembly of subprograms not being modified on the UPDATE Input Tape may be omitted by proper use of the SKIPTØ pseudo-operations.

When FAP reads a card image from the System Input Tape with SKIPTØ in its operation field, assembly is suspended and the UPDATE Input Tape is read until a card image of serialization identical to the serialization of the SKIPTØ card is found. Normal updating and assembly commence with the card of identical serialization on the UPDATE Input Tape. A card of serialization higher than that of the SKIPTØ card will not terminate the SKIPTØ operation; the serializations must be identical. Thus, assembly of a subprogram can be avoided by using a SKIPTØ card serialized with the serial number of the first card in the next subprogram to be updated. Subprograms must be updated and assembled in the order that they appear on the UPDATE Input Tape; SKIPTØ cannot be used to move the UPDATE Input Tape backward.

It is good practice to include a SKIPTØ card in the input deck for every subprogram to be updated. If the UPDATE Input Tape is positioned ready to read the card specified by the SKIPTØ card, FAP will begin assembly with that card. Inclusion of the SKIPTØ cards in all input decks makes each subprogram independent of all others. The input cards for a particular subprogram may be removed from the complete input deck without
SYSINT - System Input Tape
UDINT - UPDATE Input Tape
SYSINT=UDINT - Serialization on card from SYSINT equals serialization on card from UDINT.
SYSINT<UDINT - Serialization on card from SYSINT is less than serialization on card from UDINT.
SYSINT>UDINT - Serialization on card from SYSINT is greater than serialization on card from UDINT.

Figure 5.1a. UPDATE flow diagram.
Figure 5.1b. UPDATE flow diagram.
### 5.2 Order of Subprograms

The need to add a `SKIPTO` card in the deck for the following subprogram. The first three cards for each subprogram to be updated should be:

```
* FAP
  UPDATE  11
  SKIPTØ
```

![Serial number of the first card in the subprogram to be updated.]

Remember that the `UPDATE` Input Tape contains the unaltered, symbolic version of the CAP subprograms as contained in the listing in Appendix A. When we submit a deck to update a CAP subprogram, it is the combination of that symbolic input deck and the unaltered symbolic program on the `UPDATE` Input Tape that is assembled. When new changes are made to a subprogram, all previous desired changes to that subprogram must be included in the input deck.

### 5.3 How CAP Is Tested

If the modified version of CAP assembles successfully, it may be tested on the same computer run. To simplify this testing a special library tape is used with the FØRTRAN Monitor System. This library tape contains the execution monitor program and all of the subroutines of the CAP assembler in an unmodified, binary form. The student need only
assemble those subprograms of CAP for which changes are desired, and the library will provide the rest of the subroutines needed to complete CAP. The student must also provide a main program which calls the execution monitor program.

Once a subprogram has been modified, assembled, and checked out, it may be submitted on later runs in binary form; it need not be reassembled if no changes are to be made to it.

Let us suppose that a student has made a change to one subprogram, VAREVL, in his attempt to add division to the variable field operations. If he submits an assembly and a main program as an FMS job, the following steps will be carried out:

1. The FAP assembly will take place.

2. If the assembly is successful, the main program and the program just assembled, VAREVL, will be loaded into core memory.

3. The library will be searched for the rest of the CAP assembler and the execution monitor, and they will be loaded into core memory.

4. The CAP assembler, as modified, is then run under the execution monitor program. The input-output simulator will provide a symbolic test program for CAP to assemble. A typical symbolic program used to test CAP is shown in Appendix B.

5. When CAP finishes its assembly of the test program (or gets into a loop or stops because of the modifications), control of the computer returns to the execution monitor which prints out for debugging and comparison purposes, the following:

   a. The symbolic test program CAP worked on.

   b. The collation tape, if anything was written on it by subroutine WCT1. The collation tape is printed out in BCD.

   c. An octal postmortem of all programs which were submitted (in this case, only VAREVL and the main program).

   d. An octal postmortem of the region in core storage in which CAP was to have placed the assembled program.

In the case of the VAREVL test, it will be noted that the symbolic test program in Appendix B has in it several variable field division signs. Examination of the addresses assembled for these instructions will tell whether or not the modification worked correctly.

In case of difficulty, such as a program stop or loop, the collation tape dump is often most helpful if the stop occurred in pass one, since the tape will contain the last instruction processed correctly. Similarly, pass two loops or stops may be diagnosed by observing which instruction was the last processed and printed on the CAP assembly listing. For example, if the first instruction which does not appear on the CAP output listing is the first instruction in which division appears in the variable field, one might suspect the new VAREVL modification.

In connection with item five, listed earlier in this section, the execution monitor assumes CAP to be in an endless loop if it takes longer than five seconds to complete its assembly. The postmortem indicates the instruction location where the program was stopped. Adding one to this location will give the instruction which was next to be executed. A normal CAP assembly takes about one second on the IBM 7090 and the most complicated interaction of modifications should not extend this time by more than three seconds.

A typical CAP execution run is shown in Appendix B following assembly listings of the execution monitor subprograms. The format of the CAP assembly output and of the postmortem outputs can be seen there.
5.4 Tactics for Modifying CAP

Experience has shown that the following tactics can be helpful in making maximum use of the limited number of computer runs available for debugging modifications to CAP.

1. Some modifications are closely related to others; making the first modification allows the second to follow with but a few instructions.

2. All anticipated modifications should be submitted before the fourth or fifth run (if eight runs are available) to allow sufficient time for debugging.

3. Leave the addition of pseudo-operations which change the ILC (such as BSS) until later runs; debugging the simpler modifications in early runs. (If one of these pseudo-operations fails, the result is usually catastrophic.)

4. Observe that the point values attached to modifications are an indication of their relative difficulty. In particular, modifications to the compiler require an understanding of advanced material in Chapter 4 and should be avoided by the beginner.

5.5 The Instructor's Point of View

The material discussed in this section is of an advanced nature and may be skipped by the reader not interested in teaching CAP to a class.

The Execution Monitor

The execution monitor is a package of library subroutines called by a main program. The calling sequence to this monitor is

\[
\text{TSX} \quad \$\text{TESTS,4}
\]

The main program listed in Appendix B, which contains the above instruction, may be assembled and given to the student in binary form for submission along with his modifications. The main program also contains three words of octal 7's which prevent the student from duplicating the binary cards on an IBM 026 keypunch. Without the octal 7's the 026 may duplicate the cards incorrectly but the 7's prevent all duplication, and thus they insure against the possibility of an incorrect binary main program. Note also that the execution monitor does not return to the main program which called it, it exits to the FORTRAN Monitor System when finished testing CAP.

The execution monitor first prints a subprogram storage map of all binary and symbolic programs submitted by the student. This is done by reference to subroutine MØVIE) inserted at the time of loading by the BSS loader.* The storage map lists all subprograms found in MØVIE) from the beginning of core storage up to the subroutine TESTS, which is the first subprogram loaded from the library.

Depending on the status of sense switch one, either a core storage clock or a magnetic tape on channel B of the 7090 in combination with a data channel trap is used as a five-second timer. In the latter case, a scratch tape (tape B3 as the program is shown in Appendix B) is write selected and a sequence of data channel commands with a word count of 50000 and terminating with an IØCT command is given to channel B.

* Subroutine MØVIE) is a copy of the BSS loader table which has been moved to a position following the last subprogram loaded and given an entry point name by the BSS loader before beginning execution. This loader table consists of entry name and entry point pairs and permits a selective storage map and postmortem to be given.
Since the word transmission rate of a 729 mod IV magnetic tape is about 10,000 words per second, the data channel trap will occur in about 5 seconds if CAP has not completed its assembly and returned to the monitor by that time. This trap will restart the computer if it is at a program stop.

Other trap returns are also set by the execution monitor. A standard floating point trap interpreter is provided which changes underflow to zero and terminates the run on overflow. The select trap return is set up and the select trap enabled before calling CAP.

After these traps have been enabled, the execution monitor places in the AC the origin of the symbolic program that CAP is to assemble (50000) and calls CAP.

An I/Ø simulator package handles all calls for input and output from CAP. The input tape is simulated by a core storage buffer containing strings of card images. Subprogram FRØG is used as a buffer to hold these strings. The collation tape is also simulated using a core buffer.

Control eventually returns to the execution monitor; it returns either via the expected return from CAP, or via timer or select traps. The execution monitor prints an appropriate comment and gives a postmortem of relevant information. It then returns to the FORTRAN Monitor System with a standard system load sequence.

Miscellaneous Details About the Laboratory

If a student has made a modification which is not tested in the symbolic test program contained in subprogram PRØG, a special input/output package is used which reads card images from the System Input Tape after the student's * DATA card. All other I/Ø operations are handled in exactly the same way as in the usual I/Ø simulator package.

Each student must have the UPDATE Input Tape rewound at the beginning of his job. This rewind may be accomplished in one of several ways; perhaps the simplest is the temporary modification of the FORTRAN Monitor System to rewind the tape between jobs. An alternative might be to require that each student use theREWIND pseudo-operation in his first FAP assembly.

Making an UPDATE Input Tape

The UPDATE Input Tape used for CAP may be made with the aid of the FAP UPDATE facility. In the following discussion, since the tape is being written, it will be referred to as an UPDATE Output Tape. When making an UPDATE tape from a card deck, only an output tape is specified on the UPDATE card. For example, if the tape being written is on logical drive 11, the FAP control card would be

UPDATE ,11,,D

The D in the fourth subfield specifies that assembly is deleted, permitting the entire tape, including all subroutines, to be written with only one loading of FAP.

Since the third subfield is void, the output tape will be in blocked format. This blocked format is preferable to unblocked, as less time will be required to move the UPDATE tape when it is used later by a class. (FAP writes blocked records 16 cards to a block.)

Since assembly is deleted by the fourth subfield, regular END cards (in the subroutines being placed on the UPDATE Output Tape) will not stop FAP: the pseudo-operation ENDUP will. Following the last subprogram being placed on the UPDATE Output Tape, the UPDATE pseudo-operations ENDFIL andREWIND may be used to complete the tape.

If a student should attempt to SKIPTØ a serial number not on the UPDATE tape, FAP will stop with a comment and print the last card on the UPDATE tape. For this reason, a card with a distinctive comment such as "SKIPTØ ERRØR" may be inserted after the last subprogram written on the UPDATE tape.

Appendix A

LISTING OF THE CLASSROOM ASSEMBLY PROGRAM

This appendix consists of FAP listings of the complete Classroom Assembly Program. At the end of these listings is an assembly output produced by CAP, of a sample CAP language program. Certain conventions have been observed in these listings. The double asterisk (\(**\)) has been used as a zero element in the variable field of those instructions subject to program modification. Each subroutine begins with the pseudo-operation PCC to insure that all cards in the original subprogram appear on the listing. Since the listings are to be used as references for UPDATE modifications, the position of all control cards must be known.

<table>
<thead>
<tr>
<th>Index to Appendix A</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main program</td>
<td>48</td>
</tr>
<tr>
<td>CAP</td>
<td>50</td>
</tr>
<tr>
<td>PASS1</td>
<td>52</td>
</tr>
<tr>
<td>PASS2</td>
<td>55</td>
</tr>
<tr>
<td>VAREVL</td>
<td>61</td>
</tr>
<tr>
<td>ØPTBL</td>
<td>67</td>
</tr>
<tr>
<td>INTØP</td>
<td>69</td>
</tr>
<tr>
<td>CØMMA,</td>
<td>72</td>
</tr>
<tr>
<td>SYMSTØ,</td>
<td>76</td>
</tr>
<tr>
<td>ENDØP, PIVAR, GENØP, GNSTØ, ERASE</td>
<td>78</td>
</tr>
<tr>
<td>CØMPØP</td>
<td>81</td>
</tr>
<tr>
<td>EXPR</td>
<td>86</td>
</tr>
<tr>
<td>TERM</td>
<td>91</td>
</tr>
<tr>
<td>READ1, PRINT, WCT1, REWIND, READ2</td>
<td>95</td>
</tr>
</tbody>
</table>
MAIN PROGRAM FOR CAP.

TRANSFER VECTOR
00000 336256366477 .SETUP
00001 475131456360 PRINT
00002 232147606060 CAP
00003 256731636060 EXIT

00004 0074 00 04 00000 CALL .SETUP SETUP LIBRARY TIMER AND DUMP RETURNS.
00005 1 00000 0 00007
00006 0 00004 0 00000
00007 0074 00 04 00001
00010 0 00007 0 00031
00011 -0500 00 0 00030
00012 -074 00 04 00002
00013 -0130 00 0 00000
00014 -0763 00 0 00025
00015 -0754 00 0 00000
00016 -0774 00 4 00009
00017 -0767 00 0 00033
00020 -0763 00 0 00003
00021 2 00001 4 00017
00022 -0602 00 00047
00023 0074 00 4 00001
00024 0 00011 0 00040
00025 0 07400 4 00003
00026 1 00000 0 00030
00027 0 00201 0 00000

C0030 +000000050000 ORG OCT 50000 CAP PROGRAM ORIGIN.
00031 016060606060
00032 060606060606
00033 036352636046
00034 266023214773
00035 00225273145
00036 060216222544
00037 224370336060
00040 006060606060 RET BCI 9,0 RETURN FROM CAP, ENTRY POINT IS 00000.
00041 060606060651
00042 256364514560
00043 265146446023
00044 214773602545
00045 635170604746
00046 314563603162
00047 600000000000
00050 336060606060 END

MAIN0010
MAIN0020
MAIN0030
MAIN0040
MAIN0050
MAIN0060
MAIN0070
MAIN0080
MAIN0090
MAIN100
MAIN110
MAIN120
MAIN130
MAIN140
MAIN150
MAIN160
MAIN170
MAIN180
MAIN190
MAIN200
MAIN210
MAIN220
MAIN230
MAIN240
MAIN250
MAIN260
MAIN PROGRAM FOR CAP.
POST PROCESSOR ASSEMBLY DATA

51 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

31  BEG  10
  2  CAP  12
  30  ORG  11
  40  RET  22,  24
  3  EXIT  25
  1  PRINT  7,  23
  0  SETUP  4

NO ERROR IN ABCVE ASSEMBLY.
TIME SPENT IN FAP..  000003 IN HUNDREDTHS OF MINUTES.
SUBROUTINE CAP, CLASS ASSEMBLY PROGRAM.

TRANSFER VECTOR
00000 472162620160 PASS1
00001 472162620260 PASS2
00002 475131456360 PRINT

LINKAGE DIRECTOR
00003 000000000000
00004 232147608680

00005 0634 00 0 00200 CAP SXA RX4,4 SAVE IR4.
00006 0601 00 0 00025 STO ORG SAVE ORIGIN.
00007 0441 00 0 00034 LDI =0 CLEAR ERROR FLAGS.
00010 0074 00 0 00000 TSX #$PASS1,4 GO TO PASS1.
00011 -0054 00 00001 LFT 1 TEST FOR SYMBOL TABLE OVERFLOW.
00012 0020 00 0 00022 TRA STFL YES, GIVE DIAGNOSTIC.
00013 0604 00 0 00026 STOK STI SIND SAVE PASS1 FLAGS.
00014 0441 00 0 00034 LDI =0 CLEAR INDICATORS FOR PASS2.
00015 0500 00 0 00025 CLA ORG GET ORIGIN.
00016 0074 00 4 00001 TSX #$PASS2,4 GO TO PASS2.
00017 0442 00 0 00026 OSI SIND FORM COMPLETE ERROR FLAGS.
00020 0774 00 4 00000 RX4 AXT **,4 RESTORE IR4.
00021 0020 00 4 00001 TRA 1,4 RETURN.
00022 0074 00 4 00002 STFL TSX #$PRINT,4 COMMENT.
00023 0 00005 0 0027 PZE WSTFL,0,5 RETURN FOR PASS2 ANYWAY.
00024 0020 00 0 0013 TRA STOK RETURN FOR PASS2 ANYWAY.
00025 0 00000 0 00000 ORG PZE STORAGE FOR PROGRAM ORIGIN.
00026 0 00000 0 00000 SIND PZE STORAGE FOR SENSE INDICATORS.
00027 06270442246 WSTFL BCI 5,C SYMBOL TABLE SIZE EXCEEDED.
00030 436063212243
00031 256062317125
00032 602567232525
00033 242524336080

END

LITERALS
00034 000000000000

CAP000010
CAP000020
CAP000040
CAP000050
CAP000060
CAP000070
CAP000080
CAP000090
CAP000100
CAP000110
CAP000120
CAP000130
CAP000140
CAP000150
CAP000160
CAP000170
CAP000180
CAP000190
CAP000200
CAP000210
CAP000220
CAP000230
CAP000240
CAP000250
CAP000260
CAP000270
CAP000280
CAP000290
CAP000300
CAP000310
CAP000320
SUBROUTINE CAP, CLASS ASSEMBLY PROGRAM.
POST PROCESSOR ASSEMBLY DATA

35 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS
  5  CAP
  25  ORG  6, 15
  20  RX4  5
  26  SINO  13, 17
  22  STFL  12
  13  STCK  24
  0  PASS1 10
  1  PASS2 16
  2  PRINT 22
  27  WSTFL 23

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP.. 000003 IN HUNDREDTHS OF MINUTES.
PASS 1 OF CLASS ASSEMBLY PROGRAM.

TRANSFER VECTOR
00000 662367010600 WCT1
00001 512521240160 READ1
00002 622324566060 SCAN
00003 627044426346 SYMSTO
00004 234644442160 COMMA
00005 234644447667 COMPOP
00006 254524464760 ENDP
00007 512566314524 REWIND

LINKAGE DIRECTOR
00010 000000000000
00011 472162620160

00012 0634 00 4 00076 PASS1 SXA RX4,4 SAVE IR4.
00013 0737 00 1 00000 PAC 0,1 IR1 IS -(ILC).
00014 0020 00 0 00017 TRA RCI SKIP TAPE WRITING FOR FIRST CARD.
00015 0074 00 4 00000 NEXT TSX $WCT1,4 WRITE CARD ON COLLATION TAPE.
00016 0 00000 0 00100 PZE BUUFF **
00017 0074 00 4 00001 RCD TSX $READ1,4 READ IN NEXT CARD.
00018 0 00000 0 00100 PZE BUUFF **
00019 -0500 00 0 00101 CFLD CAL BUUFF+1 GET OP-FIELD.
00020 0074 00 4 00002 TSX $SCAN,4 COMPRESS TO RIGHT.
00021 0774 00 4 00012 AKT NPTBL,4 CHECK FOR PSEUDO-OP.
00022 -0340 00 4 00045 LAS PTBL+NPTBL,4 COMPARE WITH TABLE.
00023 0020 00 0 00027 TRA **2 NO, SKIP.
00024 0020 60 4 00046 TRA* PTBL+NPTBL+1,4 YES, EXIT.
00025 2 00002 4 00024 TXI ==3,4,2 NO, INDEX AND TRY AGAIN.
00026 -0500 00 0 00100 OP CAL BUUFF NOT A PSEUDO-OP, ASSUME OP.
00027 0074 00 4 00003 TSX $SYMSTO,4 GET SYMBOL, AND SAVE.
00028 1 77777 1 00015 TXI NEXT;1,-1 ADVANCE ILC AND RETURN.

SPACExx 2

00033 000000512544 PTBL BCI 1,000REM REMARK CARD.
00034 0020 00 0 00045 TRA REM **
00035 000000314563 BCI 1,000INT FORTRAN INTEGER.
00036 0020 00 0 00046 TRA INT **
00037 0000046236343 BCI 1,000CTL SIMPLE OCTAL.
00038 0020 00 0 00062 TRA OCTL **
00039 000023464447 BCI 1,000COMP ARITHMETIC.
00040 0020 00 0 00053 TRA COMP **
00041 000000254524 BCI 1,000END END CARD.
00042 0020 00 0 00065 TRA END **

52

PAS10010
PAS10020
PAS10040
PAS10050
PASS 1 OF CLASS ASSEMBLY PROGRAM.

00012 NPTBL EQU **-PTBL 2*(NUMBER OF PSEUDO-OPS).

00045 0020 00 0 00015 REM TRA NEXT IGNORE REMARK, RETURN.

00046 -0500 00 0 00100 INT CAL BUFF GET SYMBOL.

00047 0074 00 4 00003 TSX $SYMSTO,4 SAVE.

00050 0074 00 4 00004 TSX $COMMA,4 GO COUNT COMMAS.

00051 0 00000 0 00100 PZE BUFF **

00052 0020 00 0 00015 TRA NEXT RETURN.

00053 -0500 00 0 00100 COMP CAL BUFF GET SYMBOL.

00054 0074 00 4 00003 TSX $SYMSTO,4 SAVE.

00055 0074 00 4 00000 TSX $MCTI,4 WRITE COMP CARD ON COLLATION TAPE.

00056 0 00000 0 00100 PZE BUFF **

00057 0074 00 4 00005 TSX $COMPQ,4 GO COMPILE.

00060 0 00000 0 00100 PZE BUFF **

00061 0020 00 0 00017 TRA RCD RETURN FOR NEXT CARD.

00062 -0500 00 0 00100 OCTL CAL BUFF GET SYMBOL.

00063 0074 00 4 00003 TSX $SYMSTO,4 SAVE.

00064 1 77777 1 00015 TXI NEXT,1,-1 ADVANCE ILC AND RETURN.

00065 0074 00 4 00006 END TSX $ENODP,4 GO TO RESERVE STORAGE AND LITERALS.

00066 0074 00 4 00000 TSX $MCTI,4 WRITE END CARD ON COLLATION TAPE.

00067 0 00000 0 00100 PZE BUFF **

00070 -0500 00 0 00100 CAL BUFF GET SYMBOL.

00071 0074 00 4 00003 TSX $SYMSTO,4 SAVE.

00072 0074 00 4 00007 TSX $REWIN,4 REWIN COLLATION TAPE.

00073 0754 00 1 00000 PXA 0,1 GET FIRST LOCATION NOT USED BY PROGRAM.

00074 0737 00 4 00000 PCL 0,4 RECOMPLEMENT.

00075 0754 00 4 00000 PXA 0,4 PUT IN AC.

00076 0774 00 4 00000 RX4 AXT **,4 RESTORE IR4.

00077 0020 00 4 00001 TRA 1,4 RETURN TO CALLER.

00100 BUFF BSS 14 STORAGE FOR HOLLERITH CARD IMAGE.
116 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

30   OP
65   END  44
46   INT  36
17   RCD  14,  61
45   REM  34
76   RX4  12
100  BUFF 16,  20,  21,  30,  46,  51,  53,  56,  60,  62,  67,  70
53   COMP 42
15   NEXT 32,  45,  52,  64
62   OCTL 40
21   OFLD
33   NPTBL 24,  26,  45
2   SCAN 22
0   WCTL 15,  55,  66
4   COMMA 50
6   ENDCP 65
12  NPTBL 23,  24,  26,  45
12  PASS1
1   READ1 17
5   COMPP 57
7   REWIND 72
3   SYMS1 31,  47,  54,  63,  71

NO ERROR IN ABCVE ASSEMBLY.
*TIME SPENT IN FAP.. 000005 IN HUNDREDTHS OF MINUTES.
PASS 2 OF CLASS ASSEMBLY PROGRAM.

```
TRANSFER VECTOR
00000 464763224360 OPTBL
00001 512521240260 READ2
00002 622321456060 SCAN
00003 652151256543 VAREVL
00004 314563464760 INTOP
00005 475131456360 PRINT

00006 000000000000
00007 4721622620260

00010 0634 00 4 00136 PASS2 SXA Rx4,4 SAVE IR4.
00011 0737 00 1 00000 PAC 0,1 IRI IS -(ILC).
00012 -0500 60 0 00000 CAL* $OPTBL SETUP SEARCH FOR OP.
00013 0771 00 0 00022 ARS 18 LENGTH TO ADDRESS.
00014 0621 00 0 00037 STA OPA0X SAVE LENGTH.
00015 0361 60 0 00000 AEL* $OPTBL FIRST+LTH.
00016 0621 00 0 00040 STA OPLAS SAVE FOR LAX.
00017 0621 00 0 00047 STA OFPND SAVE FOR PICKUP.
00020 0441 00 0 00312 LDI =0 CLEAR INDICATORS.
00021 0600 00 0 00263 STZ FLGSM CLEAR TOTAL FLAGS.
00022 0442 00 0 00263 NEXT OSI FLGSM FORM COMPOSITE FLAGS.
00023 0604 00 0 00263 STI FLGSM SAVE.
00024 0441 00 0 00312 LDI =0 CLEAR INDICATORS.
00025 0634 00 1 00257 SXA LOC,1 INITIAL ILC FOR EACH CARD.
00026 0074 00 4 00001 TSX $READ2,4 READ IN NEXT CARD.
00027 0 00000 0 00274 PZE BUFF **
00030 -0500 00 0 00275 OFLD CAL BUFF+1 GET COMPRESSED OP-FIELD.
00031 0074 00 4 000002 TSX $SCAN,4 **
00032 0774 00 4 00012 AXT NPTBL,4 CHECK FOR PSEUDO-OP.
00033 -0340 00 4 00070 LAS PTBL+NPTBL,4 COMPARE WITH TABLE.
00034 0020 00 0 00036 TRA **2 NO, SKIP.
00035 0020 60 4 00071 TRA+ PTBL+NPTBL+1,4 YES, EXIT.
00036 2 00002 4 00033 TIX **3,4,2 NO, INDEX AND TRY AGAIN.
00037 0774 00 4 00000 OPA0X AXF **,4 OP, GET LENGTH OF OP-TABLE FOR SEARCH.
00040 -0340 00 4 00000 OPLAS LAS **,4 COMPARE WITH CURRENT TABLE ENTRY.
00041 0020 00 0 00043 TRA **2 NO, SKIP.
00042 1 7777 4 00047 TIX OPFND4,4-1 FOUND, INDEX AND EXIT.
00043 2 00002 4 00040 TIX **3,4,2 NO, INDEX AND TRY AGAIN.
00044 0055 00 00002 SIR 2 ILLEGAL OPCODE, SET FLAG.
00045 -0794 00 0 00000 PXO 0,0 TAKE ZERO FOR OP.
00046 0020 00 0 00050 TRA OPFND+1 SKIP PICKUP.
00047 -0500 00 4 00000 OPFNC CAL **,4 PICKUP OPCODE FROM OP-TABLE.

PAS20010
PAS20020
PAS20030
PAS20050
```
PASS 2 OF CLASS ASSEMBLY PROGRAM.

00050 0602 00 1 00000 SLW 0,1 INSERT IN ASSEMBLED PROGRAM.

00051 0074 00 4 00003 TSX $WAREVL,4 GO EVALUATE VARIABLE FIELD.

00052 0 00000 0 00274 PZE BUFF **

00053 -0602 00 1 00000 DRS 0,1 OR TO WORD IN ASSEMBLED PROGRAM.

00054 0074 00 4 00142 TSX PRNT1,4 GO PRINT ASSEMBLY LISTING.

00055 1 77777 1 00022 TXI NEXT,1,-1 RETURN.

SPACE 2
PSEUDO-OP TABLE AND TRANSFERS.

00056 000000512544 PTBL BCI 1,000REM REMARK CARD.

00057 0020 00 0 00070 TRA REP **

00058 000000314653 BCI 1,000INT FORTRAN INTEGER.

00059 0020 00 0 00072 TRA INT **

00060 000046236343 BCI 1,000OCTL SIMPLE OCTAL.

00061 0020 00 0 00076 TRA OCTL **

00062 000002346447 BCI 1,000COMP ARITHMETIC.

00063 0020 00 0 00111 TRA COMP **

00064 000000254524 BCI 1,000END END CARD.

00065 0020 00 0 00113 TRA END **

00066 0000000254624 NPTBL EQU *-PTBL 2*(NUMBER OF PSEUDO-OPS).

SPACE 2
PSEUDO-OPS.

00070 0074 00 4 00173 REM TSX PRNT2,4 PRINT REMARK.

00071 0020 00 0 00022 TRA NEXT RETURN.

00072 0074 00 4 00004 INT TSX $INTOP,4 DO INTEGER CONVERSION.

00073 0 00000 0 00274 PZE BUFF **

00074 0074 00 4 00142 TSX PRNT1,4 GO PRINT ASSEMBLY LISTING.

00075 0020 00 0 00022 TRA NEXT RETURN.

00076 -0754 00 0 00000 OCTL PXD 0,0 SIMPLE OCTAL, CLEAR AC.

00077 0774 00 2 00002 AKT 2,2 2 WORDS.

00078 0774 00 4 00006 OLP AKT 6,4 6 CHARACTERS PER WORD.

00079 0560 00 0 00300 L010 LOU BUFF+2*2,2 GET WORD.

00080 0773 00 0 00003 RQL 3 BCI-OCTAL CONVERSION.

00081 0763 00 0 00003 LGL 3 **

00082 2 00001 4 00102 TIX =$-2,4,1 COUNT CHARACTERS.

00083 2 00001 2 00100 TIX OLP,2,1 COUNT WORDS.

00084 0602 00 1 00000 SLW 0,1 INSERT IN ASSEMBLED PROGRAM.

00085 0074 00 4 00142 TSX PRNT1,4 GO PRINT ASSEMBLY LISTING.

00086 1 77777 1 00022 TXI NEXT,1,-1 INDEX AND RETURN.

00087 0074 00 4 00173 COMP TSX PRNT2,4 PRINT COMP AS A REMARK.

00088 0020 00 0 00022 TRA NEXT RETURN.

00089 0074 00 4 00003 END TSX $WAREVL,4 EVALUATE VARIABLE FIELD.

00090 0 00000 0 00274 PZE BUFF **

00091 0601 00 0 00260 STD EPNT SAVE AS ENTRY POINT.
PASS 2 OF CLASS ASSEMBLY PROGRAM.

00116  0074  00  4  00204  TSX  FLGS,4  GET FLGS.  PAS20970
00117  0602  00  0  00267  SLW  PBUFF  PLACE IN PBUFF.  PAS20980
00120  0500  00  0  00260  CLA  EPNT  CONVERT ENTRY POINT.  PAS20990
00121  0074  00  4  00225  TSX  OCTA,4  **  PAS21000
00122  0560  00  0  00312  LDQ  =0  CLEAR MQ.  PAS21010
00123  -0765  00  0  00222  LGR  1B  SHIFT TO POSITION.  PAS21020
00124  -0501  00  0  00316  ORA  =H  000  INSERT BLANKS LEFT.  PAS21030
00125  0602  00  0  00272  SLW  PBUFF+3  INSERT IN PBUFF.  PAS21040
00126  -0130  00  0  00000  XCL  **  GET RIGHT HALF.  PAS21050
00127  -0501  00  0  00315  ORA  =H000  INSERT BLANKS RIGHT.  PAS21060
00130  0602  00  0  00273  SLW  PBUFF+4  INSERT IN PBUFF.  PAS21070
00131  -0500  00  0  00317  CAL  =H  BLANK OUT REST OF PBUFF.  PAS21080
00132  0602  00  0  00270  SLW  PBUFF+1  **  PAS21090
00133  0602  00  0  00271  SLW  PBUFF+2  **  PAS21100
00134  0074  00  4  00005  TSX  $PRINT,4  GO PRINT.  PAS21110
00135  0  00023  0  00267  PZE  PBUFF,0,19  **  PAS21120
00136  0774  00  4  00000  RX4  **,4  restore IR4.  PAS21130
00137  0500  00  0  00260  CLA  EPNT  GET ENTRY.  PAS21140
00140  0442  00  0  00263  OSI  FLGS,4  GET TOTAL ERROR FLAGS.  PAS21150
00141  0020  00  4  00001  TRA  1,H  RETURN.  PAS21160

SPACE 2
PRINT ROUTINES.

00142  0634  00  4  00171  PRNT1  SXA  PI4X,4  SAVE IR4.  PAS21170
00143  0074  00  4  00204  TSX  FLGS,4  GET ERROR FLGS.  PAS21180
00144  0602  00  0  00267  SLW  PBUFF  PLACE IN PBUFF.  PAS21190
00145  0535  00  4  00257  LAC  LOC,4  GET *(1LC).  PAS21200
00146  0754  00  4  00000  PXA  0,4  **  PAS21210
00147  0074  00  4  00225  TSX  OCTA,4  CONVERT OCTAL ADDRESS.  PAS21220
00150  0602  00  0  00270  SLW  PBUFF+1  PLACE IN PBUFF.  PAS21230
00151  0534  00  4  00257  LAX  LOC,4  GET *(1LC).  PAS21240
00152  -0500  00  4  00000  CAL  0,4  GET SELECTED WORD.  PAS21250
00153  0074  00  4  00236  TSX  OCTR,4  CONVERT OCTAL WORD.  PAS21260
00154  -0600  00  0  00262  STQ  RHQCT  SAVE RIGHT HALFW.  PAS21270
00155  -0765  00  0  00222  LGR  1B  SHIFT TO POSITION.  PAS21280
00156  -0501  00  0  00316  ORA  =H  000  INSERT BLANKS LEFT.  PAS21290
00157  0602  00  0  00271  SLW  PBUFF+2  PLACE IN PBUFF.  PAS21300
00160  -0600  00  0  00272  STQ  PBUFF+3  **  PAS21310
00161  -0500  00  0  00262  CAL  RHQCT  GET RIGHT HALFW.  PAS21320
00162  0560  00  0  00312  LDQ  =0  ZERO MQ.  PAS21330
00163  -0765  00  0  00222  LGR  1B  SHIFT TO POSITION.  PAS21340
00164  -0130  00  0  00000  XCL  **  PLACE IN AC.  PAS21350
00165  -0501  00  0  00315  ORA  =H000  INSERT BLANKS RIGHT.  PAS21360
00166  0602  00  0  00273  SLW  PBUFF+4  PLACE IN PBUFF.  PAS21370
00167  0074  00  4  00005  TSX  $PRINT,4  GO PRINT.  PAS21380
00170  0  00023  0  00267  PZE  PBUFF,0,19  **  PAS21390
00171  0774  00  4  00000  PI4X  AX,4  RESTORE IR4.  PAS21400
00172  0020  00  4  00001  TRA  1,H  RETURN.  PAS21410

00173  0634  00  4  00202  PRNT2  SXA  P2X4,4  SAVE IR4.  PAS21420
00174  0774  00  4  00005  AX,4  5,4  BLANK OUT PBUFF TO PBUFF+4.  PAS21430
00175  -0500  00  0  00317  CAL  =H  **  PAS21440
00176  0602  00  0  00274  SLW  PBUFF+5,4  **  PAS21450
00177  2  00001  4  00176  TIX  =-1,H  **  PAS21460
<table>
<thead>
<tr>
<th>Address</th>
<th>Mnemonic</th>
<th>8-bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00200</td>
<td>TSX</td>
<td>08</td>
<td>TSX $PRINT,4</td>
</tr>
<tr>
<td>00201</td>
<td>$PRINT,4</td>
<td>04</td>
<td>GO PRINT.</td>
</tr>
<tr>
<td>00202</td>
<td>P2X4</td>
<td>06</td>
<td>P2X4</td>
</tr>
<tr>
<td>00203</td>
<td>TRA</td>
<td>11</td>
<td>TRA 1,4</td>
</tr>
</tbody>
</table>

### SPACE 2

#### BCI CONVERSION ROUTINES.

<table>
<thead>
<tr>
<th>Address</th>
<th>Mnemonic</th>
<th>8-bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00204</td>
<td>FLX4,4</td>
<td>06</td>
<td>FLX4,4</td>
</tr>
<tr>
<td>00205</td>
<td>TOV</td>
<td>07</td>
<td>TOV ++1</td>
</tr>
<tr>
<td>00206</td>
<td>PIA</td>
<td>08</td>
<td>PIA</td>
</tr>
<tr>
<td>00207</td>
<td>LGR</td>
<td>09</td>
<td>LGR NFLGS</td>
</tr>
<tr>
<td>00208</td>
<td>CAL</td>
<td>0A</td>
<td>CAL NFLGS4</td>
</tr>
<tr>
<td>00209</td>
<td>TQP</td>
<td>0B</td>
<td>TQP ++3</td>
</tr>
<tr>
<td>00210</td>
<td>ALS</td>
<td>0C</td>
<td>ALS 6</td>
</tr>
<tr>
<td>00211</td>
<td>ACL</td>
<td>0D</td>
<td>ACL TFLGS+NFLGS,4</td>
</tr>
<tr>
<td>00212</td>
<td>RGL</td>
<td>0E</td>
<td>RGL 1</td>
</tr>
<tr>
<td>00213</td>
<td>TIX</td>
<td>0F</td>
<td>TIX =-4,4,1</td>
</tr>
<tr>
<td>00214</td>
<td>LDQ</td>
<td>10</td>
<td>LDQ =H</td>
</tr>
<tr>
<td>00215</td>
<td>LVL</td>
<td>11</td>
<td>LVL</td>
</tr>
<tr>
<td>00216</td>
<td>TNO</td>
<td>12</td>
<td>TNO =-1</td>
</tr>
<tr>
<td>00217</td>
<td>FLX4</td>
<td>13</td>
<td>FLX4</td>
</tr>
<tr>
<td>00218</td>
<td>TRA</td>
<td>14</td>
<td>TRA 1,4</td>
</tr>
</tbody>
</table>

#### OCTA SQA

<table>
<thead>
<tr>
<th>Address</th>
<th>Mnemonic</th>
<th>8-bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00225</td>
<td>OAX4,4</td>
<td>01</td>
<td>OAX4,4</td>
</tr>
<tr>
<td>00226</td>
<td>LGR</td>
<td>02</td>
<td>LGR 13</td>
</tr>
<tr>
<td>00227</td>
<td>CAL</td>
<td>03</td>
<td>CAL =H0000</td>
</tr>
<tr>
<td>00228</td>
<td>LVL</td>
<td>04</td>
<td>LVL</td>
</tr>
<tr>
<td>00229</td>
<td>TIX</td>
<td>05</td>
<td>TIX =-2,4,1</td>
</tr>
<tr>
<td>00230</td>
<td>OAX4</td>
<td>06</td>
<td>OAX4</td>
</tr>
<tr>
<td>00231</td>
<td>TRA</td>
<td>07</td>
<td>TRA 1,4</td>
</tr>
</tbody>
</table>

#### OCTW SQA

<table>
<thead>
<tr>
<th>Address</th>
<th>Mnemonic</th>
<th>8-bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00236</td>
<td>DWX4,4</td>
<td>08</td>
<td>DWX4,4</td>
</tr>
<tr>
<td>00237</td>
<td>PFDX</td>
<td>09</td>
<td>PFDX 0,0</td>
</tr>
<tr>
<td>00238</td>
<td>AXS</td>
<td>10</td>
<td>AXS 6,4</td>
</tr>
<tr>
<td>00239</td>
<td>LVL</td>
<td>11</td>
<td>LVL</td>
</tr>
<tr>
<td>00240</td>
<td>TIX</td>
<td>12</td>
<td>TIX =-2,4,1</td>
</tr>
<tr>
<td>00241</td>
<td>PXD</td>
<td>13</td>
<td>PXD 0,0</td>
</tr>
<tr>
<td>00242</td>
<td>AXS</td>
<td>14</td>
<td>AXS 6,4</td>
</tr>
<tr>
<td>00243</td>
<td>LVL</td>
<td>15</td>
<td>LVL</td>
</tr>
<tr>
<td>00244</td>
<td>LMC</td>
<td>16</td>
<td>LMC</td>
</tr>
<tr>
<td>00245</td>
<td>TIX</td>
<td>17</td>
<td>TIX =-2,4,1</td>
</tr>
<tr>
<td>00246</td>
<td>PXD</td>
<td>18</td>
<td>PXD 0,0</td>
</tr>
<tr>
<td>00247</td>
<td>AXS</td>
<td>19</td>
<td>AXS 6,4</td>
</tr>
<tr>
<td>00248</td>
<td>LVL</td>
<td>20</td>
<td>LVL</td>
</tr>
<tr>
<td>00249</td>
<td>TIX</td>
<td>21</td>
<td>TIX =-2,4,1</td>
</tr>
<tr>
<td>00250</td>
<td>PXD</td>
<td>22</td>
<td>PXD 0,0</td>
</tr>
<tr>
<td>00251</td>
<td>AXS</td>
<td>23</td>
<td>AXS 6,4</td>
</tr>
<tr>
<td>00252</td>
<td>LMC</td>
<td>24</td>
<td>LMC</td>
</tr>
<tr>
<td>00253</td>
<td>TIX</td>
<td>25</td>
<td>TIX =-2,4,1</td>
</tr>
<tr>
<td>00254</td>
<td>PXD</td>
<td>26</td>
<td>PXD 0,0</td>
</tr>
<tr>
<td>00255</td>
<td>AXS</td>
<td>27</td>
<td>AXS 6,4</td>
</tr>
<tr>
<td>00256</td>
<td>TRA</td>
<td>28</td>
<td>TRA 1,4</td>
</tr>
</tbody>
</table>
PASS 2 OF CLASS ASSEMBLY PROGRAM.

SPACE 2

STORAGE AND CONSTANTS.

00257 0 00000 0 00000 LOC PZE
00260 0 00000 0 00000 EPNT PZE
00261 0 00000 0 00000 LHOCT PZE
00262 0 00000 0 00000 RhOCT PZE
00263 0 00000 0 00000 FLGSM PZE

00264 0000000000025 TFLGS SYN # TABLE OF ERROR FLAGS.
00265 0000000000046 BCI 1,00000 SI BIT 33.
00266 0000000000064 BCI 1,00000U SI BIT 34.

00003 NFLGS EQU *=--TFLGS NUMBER OF ERROR FLAGS.

00267 PBUFF BSS 19 PRINT BUFFER.
00274 BUFF SYN PBUFF+5 START OF CARD IMAGE BUFFER.

END

LITERALS

00312 00000000000000
00313 00000000000060
00314 00000000000160
00315 00000060606060
00316 60606060606060
00317 60606060606060
REFERENCES TO DEFINED SYMBOLS

113 END 67
72 INT 61
257 LCC 25, 145, 151
100 OLP 165
70 REM 57
136 RX4 10
274 BUFF 27, 30, 52, 73, 101, 114, 312
111 COMP 65
260 EPNT 115, 120, 137
223 FLX4 204
22 NEXT 55, 71, 75, 110, 112
234 OAX4 225
225 OCTA 121, 147
76 OCTL 63
236 OCTW 153
30 OFLD
255 OWX4 236
171 PIX4 142
202 PZ4 173
56 PTBL 33, 35, 70
2 SCAN 31
204 FLAGS 116, 143
263 FLGSM 21, 22, 23, 140
4 INTOP 72
261 LHQT 245, 254
3 NFLGS 207, 211, 214, 267
12 NPTBL 32, 33, 35, 70
37 OPAX 14
47 OVFND 17, 42, 46
40 OPLAS 16
0 OPLB 12, 15
10 PASS2
267 PBUF 117, 125, 130, 132, 133, 135, 144, 150, 157, 160, 166, 170, 176, 201, 312
5 PRINT 134, 167, 200
142 PRNT1 54, 74, 107
173 PRNT2 70, 111
1 REAC 26
262 RHQT 154, 161
264 TFLGS 214, 264, 267
3 VAREVL 51, 113

NO ERROR IN ABOVE ASSEMBLY.
TIME SPENT IN FAP.. 000009 IN HUNDREDS OF MINUTES.
**SVAREVL, SCAN AND EVALUATE VARIABLE FIELD OF CAP CARD.**

<table>
<thead>
<tr>
<th>PGC</th>
<th>COUNT</th>
<th>222</th>
<th>VAREVL</th>
<th>BINARY CARD LABEL.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBL</td>
<td>VAREVL</td>
<td>ENTRY</td>
<td>RVREVL</td>
<td>EVALUATE FIELDS BETWEEN COMMAS.</td>
</tr>
<tr>
<td>ENTR</td>
<td>VAREVL</td>
<td>ENTRY</td>
<td>RVREVL</td>
<td>EVALUATE FIRST FIELD.</td>
</tr>
</tbody>
</table>

**TRANSFER VECTOR**
00000 627042272563 SYMGET
00001 622165256060 SAVE
00002 644562216525 UNSAVE

**LINKAGE DIRECTOR**
00003 0000000000000
00004 516651256543

00005 0634 00 4 00030 VAREVL SXA RX4,4 SAVE IRS.
00006 0634 00 2 00031 SXA RX2,2 **
00007 0634 00 1 00032 SXA RX1,1 **
00010 -0500 00 4 00001 CAL 1,4 GET BUFFER ADDRESS.
00011 0361 00 0 00254 ACL *12 BUFF+12.
00012 0621 00 0 00563 STA LDQ SAVE FOR PICKUP.
00013 0600 00 0 00245 STZ TEDF RESET EOF MARK.
00014 0600 00 0 00246 STZ TCM RESET COMMA MARK.
00015 -0760 00 0 00114 SIT 1 TURN OFF LIGHT 1.
00016 0761 00 0 00000 NDP **
00017 0774 00 2 00012 AXT 10,2 COUNT 10 WORDS.
00020 0774 00 1 00007 AXT 7,1 COUNT 6 CHARACTERS.
00021 0560 60 0 00063 LDQ* LDQ GET FIRST WORD OF VARIABLE FIELD.
00022 -0600 00 1 00252 STQ MQ SAVE FOR EVAL.
00023 0704 00 4 00052 GEVAL TSX **
00024 0734 00 4 00000 PAX 0,4 GO EVALUATE FIELD.
00025 0120 00 0 00207 TPL **2 PLUS OR MINUS.
00026 0737 00 4 00000 PA0 0,4 MINUS, FORM 2S COMPLEMENT.
00027 0754 00 4 00000 PXA 0,4 FINAL RESULT IN A1ACI.
00030 0774 00 4 00000 RX4 AXT **,4 RESTORE IRS.
00031 0774 00 2 00000 RX2 AXT **,2 **
00032 0774 00 1 00000 RX1 AXT **,1 **
00033 0020 00 4 00002 TRA 2,4 RETURN WITH RESULT IN AC.

**SPACE 2**

**RE-ENTRY TO EVALUATE MULTIPLE FIELDS.**

00034 -0754 00 0 00000 RVREVL PXD 0,0 CLEAR AC.
00035 -0520 00 0 00246 NZT TCCM CHECK FOR COMMA ENCOUNTERED.
00036 0020 00 0 00001 **
00037 0600 00 0 00245 STZ TCCM RESET EOF MARK.
00040 0600 00 0 00246 STZ TCM RESET COMMA MARK.
00041 -0760 00 0 00141 SIT 1 TURN OFF LIGHT 1.
00042 0761 00 0 00000 NDP **
00043 1 00001 4 00044 TXI **1,4,1 DECREASE CALL LOCATION BY ONE.
00044 0634 00 4 00030 SXA RX4,4 SAVE IRS.
00045 0634 00 2 00031 SXA RX2,2 **
00046 0634 00 1 00032 SXA RX1,1 **
00047 0774 00 1 00000 REX1 AXT **,1 RESTORE IRS FOR POSITION IN FIELD.
00050 0774 00 2 00000 REX2 AXT **,2 **

VEVLO010
VEVLO020
VEVLO030
VEVLO050
VEVLO060

VEVLO070
VEVLO080
VEVLO090
VEVLO100
VEVLO110
VEVLO120
VEVLO130
VEVLO140
VEVLO150
VEVLO160
VEVLO170
VEVLO180
VEVLO190
VEVLO200
VEVLO210
VEVLO220
VEVLO230
VEVLO240
VEVLO250
VEVLO260
VEVLO270
VEVLO280
VEVLO290
VEVLO300
VEVLO310
VEVLO320
VEVLO330
VEVLO340
VEVLO350
VEVLO360
VEVLO370
VEVLO380
VEVLO390
VEVLO400
VEVLO410
VEVLO420
VEVLO430
VEVLO440
VEVLO450
$VAREVL: SCAN AND EVALUATE VARIABLE FIELD OF CAP CARD.

00051 0020 00 0 00023 TRA GEVAL GO EVALUATE THIS FIELD.

SPACE 2 EVALUATION SUBROUTINE, RECURSIVELY DEFINED.

00052 0634 00 0 00243 EVAL SXA EVX4,4 SAVE IR.
00053 0600 00 0 00241 STZ SUM INITIALIZE REGISTERS.
00054 0600 00 0 00242 STZ TEAM " "
00055 0600 00 0 00251 STZ VAL " "
00056 0600 00 0 00250 STZ SYM RESET SYM.
00057 0774 00 0 00255 AXT NPL,4 SET LBKCH TO PLUS.
00058 0634 00 0 00244 SXA LRBKCH,4 " "
00059 0020 00 0 00100 TRA RSCAN GO TO SCANNER.
00061 0774 00 1 00006 SCAN AX 6,1 COUNT 6 CHAR.
00063 0560 00 0 00000 LDQ LDQ **,2 PICKUP NEXT WORD, ADDRESS IS BUFF+12.
00064 -0754 00 0 00000 CHAR PXD 0,0 CLEAR AC.
00065 -0763 00 0 00006 STQ MQ SAVE MQ.
00066 -0600 00 0 00252 AXT NEK,4 COMPARE WITH LIST OF BREAKS.
00067 0774 00 0 00255 AXT NEK,4 " "
00070 -0340 00 0 00136 LAS TABBK+NBK,4 " "
00071 0020 00 0 00073 TRA **2 NOT THIS ONE, SKIP.
00072 0020 00 0 00136 TRA BKCH BREAK FOUND, EXIT.
00073 2 0000 4 00070 TIX **3,4,3 NOT THIS ONE, INDEX AND TRY AGAIN.
00074 -0765 00 0 00006 LGX 6 NOT A BREAK, BUILD SYMBOL.
00075 -0500 00 0 00250 CAL SYM " "
00076 -0763 00 0 00006 LGX 6 " "
00077 0602 00 0 00250 SLW SYM SAVE PARTIAL SYMBOL.
00100 0520 00 0 00445 RSCAN ZEF TEOF TEST FOR END-OF-FIELD.
00101 0020 00 0 00106 TRA EOFB YES, EXIT TO RPAR SECTION.
00102 0560 00 0 00252 LDQ MQ NO, RESTORE MQ.
00103 2 0000 1 00064 TIX CHAR,1,1 COUNT CHAR.
00104 2 0000 1 00062 TIX SCAN,2,1 COUNT WORDS.
00105 0625 00 0 00245 STL TEFCH END-OF-FIELD REACHED, APPEND AS
00106 0774 00 4 00003 EOFB AXT NRPAR,4 MANY RPAR AS NECESSARY.
00107 -0500 00 0 00255 CAL "="(H300C0) " "
00110 0020 00 0 00136 TRA BKCH GO TO BREAK.

SPACE 2 TABLE OF BREAKS.

01101 TABBK SYN

0111 0000000000020 CPL BCI 1,00000+ PLUS.
01112 0020 00 0 00172 TRA LPL " "
01113 0020 00 0 00210 TRA RPL " "
01114 000000000040 BCI 1,0000- MINUS.
01115 0020 00 0 00175 TRA LMI " "
01116 0020 00 0 00210 TRA RMI " "
01117 000000000054 BCI 1,0000+ STAR.
01118 0020 00 0 00200 TRA LST " "
01119 0020 00 0 00215 TRA RST " "
01120 0000 60 0 00123 HTR* " " SHOULD NEVER GET HERE.
$VAREVL, SCAN AND EVALUATE VARIABLE FIELD OF CAP CARD.

00124   0020 00 0 00237  TRA BLANK **
00125   00000000000000003  BCI 1,00000, COMMA.
00126   0000 60 0 00126  HTR* 1,0000000000000000
00127   0020 00 0 00231  TRA RCOM **
00130   00000000000000004  BCI 1,0000000000000000
00131   0000 60 0 00131  HTR* 1,0000000000000000
00132   0020 00 0 00216  TRA LPAR **
00133   00000000000000004  CRPAR BCI 1,0000000000000000
00134   0000 60 0 00134  HTR* 1,0000000000000000
00135   0020 00 0 00225  TRA RPAR **

00025  NBK EQU *=TABBK NUMBER OF BREAK CHARACTERS.
00003  NRPAR EQU *=CRPAR BREAK NUMBER OR RPAR.
00025  NPL EQU *=CPL BREAK NUMBER OF PLUS.

SPACE 2
BREAK CHARACTER SECTION.

00136   0634 00 4 00247  BKCH SXA RBCX,4 SAVE NUMBER OF RIGHT BREAK.
00137   -0340 00 0 00256  LAS =M00000 CHECK FOR LPAR.
00140   0020 00 0 00142  TRA ++2 NO, SKIP.
00141   0020 00 0 00216  TRA LPAR YES, GO TO IT.
00142   0520 00 0 00251  ZET VAL EXPRESSION, SYMBOL, OR NUMBER.
00143   0020 00 0 00167  TRA LKB EXPRESSION, NO SYMBOL TO CONVERT.
00144   -0500 00 0 00250  CAL SYM SYMBOL OR NUMBER.
00145   -0320 00 0 00257  ANA += NUMBERS HAVE NO ZONE.
00146   0100 00 0 00153  TZE NUM NUMBER, GO CONVERT.
00147   -0500 00 0 00250  CAL SYM SYMBOL, GET VALUE.
00150   0074 00 4 00000  TSX $SYMGET,4...
00151   0601 00 0 00251  STO VAL SAVE VALUE.
00152   0020 00 0 00167  TRA LKB EXIT TO LKB.
00153   0560 00 0 00250  NUM LDQ SYM NUMBER, UNSIGNED.
00154   0774 00 4 00006  AXT 6,4 COUNT 6 DIGITS.
00155   -0754 00 0 00000  NLORD PXD 0,0 CLEAR AC.
00156   -0763 00 0 00000  LGL 6 GET DIGIT.
00157   0601 00 0 00253  STO DIG SAVE.
00160   0500 00 0 00251  CLA VAL PROGRAMED IG*VAL.
00161   0767 00 0 00002  ALS 2 4*VAL.
00162   0400 00 0 00251  ADD VAL 4*VAL+VAL.
00163   0767 00 0 00001  ALS 1 2*4*VAL+VAL+10*VAL.
00164   0400 00 0 00253  ADD DIG ADD THIS DIGIT.
00165   0601 00 0 00251  STO VAL SAVE PARTIAL RESULT.
00166   2 0001 4 00155  TIX NLORD,4,1 COUNT DIGITS CONVERTED.

SPACE 2
LEFT BREAK SECTION.

00167   0600 00 0 00250  LKB STZ SYM LEFT BREAK, RESET SYM.
00170   0534 00 4 00244  LXA LBCX,4 GET NUMBER OF LEFT BREAK.
00171   0020 00 0 00137  TRA TABBK=NBK+1,4 GO TO LEFT BREAK.
00172   0500 00 0 00251  LPL CLA VAL +, TERM=VAL.
00173   0601 00 0 00242  STO TEAM **
$VAREVL, SCAN AND EVALUATE VARIABLE FIELD OF CAP CARD.

<table>
<thead>
<tr>
<th>00174</th>
<th>0020 00  0 00204</th>
<th>TRA</th>
<th>RBK</th>
<th>GO TO RIGHT BREAK.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00175</td>
<td>0502 00  0 00251</td>
<td>LMI</td>
<td>CLS</td>
<td>VAL</td>
</tr>
<tr>
<td>00176</td>
<td>0601 00  0 00242</td>
<td>STO</td>
<td>TERM</td>
<td>**</td>
</tr>
<tr>
<td>00177</td>
<td>0020 00  0 00204</td>
<td>TRA</td>
<td>RBK</td>
<td>GO TO RIGHT BREAK.</td>
</tr>
<tr>
<td>00200</td>
<td>0560 00  0 00251</td>
<td>LST</td>
<td>LDQ</td>
<td>VAL</td>
</tr>
<tr>
<td>00201</td>
<td>0200 00  0 00242</td>
<td>MPY</td>
<td>TERM</td>
<td>**</td>
</tr>
<tr>
<td>00202</td>
<td>-0600 00  0 00242</td>
<td>STQ</td>
<td>TERM</td>
<td>**</td>
</tr>
<tr>
<td>00203</td>
<td>0020 00  0 00204</td>
<td>TRA</td>
<td>RBK</td>
<td>GO TO RIGHT BREAK.</td>
</tr>
</tbody>
</table>

**SPACE 2**

**RIGHT BREAK SECTION.**

<table>
<thead>
<tr>
<th>00204</th>
<th>0534 00  0 00247</th>
<th>RBK</th>
<th>LXN</th>
<th>RBKCH,*4</th>
</tr>
</thead>
<tbody>
<tr>
<td>00205</td>
<td>0634 00  0 00244</td>
<td>SXA</td>
<td>LBKCH,*4</td>
<td>THIS IS NEXT LEFT BREAK.</td>
</tr>
<tr>
<td>00206</td>
<td>0600 00  0 00251</td>
<td>STZ</td>
<td>VAL</td>
<td>RESET VAL.</td>
</tr>
<tr>
<td>00207</td>
<td>0020 00  0 00140</td>
<td>TRA</td>
<td>TABBK+NBK+2,4</td>
<td>GO TO RIGHT BREAK.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>00210</th>
<th>0500 00  0 00241</th>
<th>RPL</th>
<th>CLA</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>00211</td>
<td>0400 00  0 00242</td>
<td>ADD</td>
<td>TERM</td>
<td>**</td>
</tr>
<tr>
<td>00212</td>
<td>0601 00  0 00241</td>
<td>STO</td>
<td>SUM</td>
<td>**</td>
</tr>
<tr>
<td>00213</td>
<td>0600 00  0 00242</td>
<td>STZ</td>
<td>TERM</td>
<td>RESET TERM.</td>
</tr>
<tr>
<td>00214</td>
<td>0020 00  0 00100</td>
<td>TRA</td>
<td>RSCAN</td>
<td>RESUME SCAN.</td>
</tr>
</tbody>
</table>

**00215** | 0020 00  0 00100 | RST     | TRA     | RSCAN             |

**SPACE 2**

**LPAR, RPAT, AND EOF SECTION.**

<table>
<thead>
<tr>
<th>00216</th>
<th>0074 00  0 00001</th>
<th>LPAR</th>
<th>TSX</th>
<th>$SAVE,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>00217</td>
<td>0 00004 0 00241</td>
<td>PZE</td>
<td>SUM,*0,4</td>
<td>**</td>
</tr>
<tr>
<td>00220</td>
<td>0074 00  0 00052</td>
<td>TSX</td>
<td>EVALL,*4</td>
<td>CALL SELF.</td>
</tr>
<tr>
<td>00221</td>
<td>0601 00  0 00251</td>
<td>STO</td>
<td>VAL</td>
<td>RESULT IS VAL.</td>
</tr>
<tr>
<td>00222</td>
<td>074 00  0 00002</td>
<td>TSX</td>
<td>SUMSAVE,*4</td>
<td>RESTORE REGISTERS.</td>
</tr>
<tr>
<td>00223</td>
<td>0 00004 0 00241</td>
<td>PZE</td>
<td>SUM,*0,4</td>
<td>**</td>
</tr>
<tr>
<td>00224</td>
<td>0020 00  0 00100</td>
<td>TRA</td>
<td>RSCAN</td>
<td>RESUME SCAN.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>00225</th>
<th>0500 00  0 00241</th>
<th>RPAR</th>
<th>CLA</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>00226</td>
<td>0400 00  0 00242</td>
<td>ADD</td>
<td>TERM</td>
<td>ADD IN CURRENT TERM</td>
</tr>
<tr>
<td>00227</td>
<td>0534 00  0 00243</td>
<td>LXA</td>
<td>EVX,*4,4</td>
<td>RESTORE IRA.</td>
</tr>
<tr>
<td>00230</td>
<td>0020 00  0 00001</td>
<td>TRA</td>
<td>1,*4</td>
<td>RETURN TO CALLER.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>00231</th>
<th>-0625 00  0 00246</th>
<th>RCOM</th>
<th>STL</th>
<th>TCOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>00232</td>
<td>-0625 00  0 00245</td>
<td>STL</td>
<td>TEOF</td>
<td>SET MARK FOR COMMA ENCOUNTERED.</td>
</tr>
<tr>
<td>00233</td>
<td>0760 00  0 00141</td>
<td>SLN</td>
<td>EXP,1</td>
<td>EXTERNAL MARK FOR COMMA ENCOUNTERED.</td>
</tr>
<tr>
<td>00234</td>
<td>0634 00  0 00047</td>
<td>Sxa</td>
<td>REEX,*1</td>
<td>SAVE IRs FOR RE-SCAN.</td>
</tr>
<tr>
<td>00235</td>
<td>0634 00  0 00050</td>
<td>Sxa</td>
<td>REEX,*2</td>
<td>**</td>
</tr>
<tr>
<td>00236</td>
<td>0020 00  0 00225</td>
<td>TRA</td>
<td>EOF</td>
<td>GO TO ECF SECTION.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>00237</th>
<th>-0625 00  0 00245</th>
<th>BLANK</th>
<th>STL</th>
<th>TEOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>00240</td>
<td>0020 00  0 00225</td>
<td>TRA</td>
<td>EOF</td>
<td>**</td>
</tr>
</tbody>
</table>
$VAREVL, SCAN AND EVALUATE VARIABLE FIELD OF CAP CARD.

00225  EOF  SYN  RPAR  SAME AS RPAR.

SPACE 2
STORAGE AREA FOR SAVE.

00241  0  00000  0  00000  SUM  PZE  **
00242  0  00000  0  00000  TERM  PZE  **
00243  0  00000  0  00000  EVX4  PZE  **
00244  0  00000  0  00000  LBKCH  PZE  **

SPACE 2
TEMPORARY STORAGE.

00245  0  00000  0  00000  TEDF  PZE  **
00246  0  00000  0  00000  TCM  PZE  **
00247  0  00000  0  00000  RBKCF  PZE  **
00250  0  00000  0  00000  SYM  PZE  **
00251  0  00000  0  00000  VAL  PZE  **
00252  0  00000  0  00000  MQ  PZE  **
00253  0  00000  0  00000  CIG  PZE  **

END

LITERALS
00254  0000000000014
00255  0000000000034
00256  0000000000074
00257  6060606060606
REFERENCES TO DEFINED SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQ</td>
<td>22, 66, 102</td>
</tr>
<tr>
<td>CPL</td>
<td>136</td>
</tr>
<tr>
<td>DIG</td>
<td>137, 164</td>
</tr>
<tr>
<td>ECQ</td>
<td>236, 240, 241</td>
</tr>
<tr>
<td>LBK</td>
<td>143, 132</td>
</tr>
<tr>
<td>LEQ</td>
<td>12, 21</td>
</tr>
<tr>
<td>LWI</td>
<td>115</td>
</tr>
<tr>
<td>LPL</td>
<td>112</td>
</tr>
<tr>
<td>LST</td>
<td>120</td>
</tr>
<tr>
<td>NBD</td>
<td>67, 70, 136, 171, 207</td>
</tr>
<tr>
<td>NPL</td>
<td>57, 136</td>
</tr>
<tr>
<td>NUM</td>
<td>146</td>
</tr>
<tr>
<td>RBK</td>
<td>174, 177, 203</td>
</tr>
<tr>
<td>RM1</td>
<td>116, 215</td>
</tr>
<tr>
<td>RPL</td>
<td>113, 215</td>
</tr>
<tr>
<td>RST</td>
<td>121</td>
</tr>
<tr>
<td>RX1</td>
<td>7, 46</td>
</tr>
<tr>
<td>RX2</td>
<td>6, 45</td>
</tr>
<tr>
<td>RX4</td>
<td>5, 44</td>
</tr>
<tr>
<td>SUM</td>
<td>53, 210, 212, 217, 223, 225</td>
</tr>
<tr>
<td>SYM</td>
<td>56, 75, 77, 144, 147, 153, 167</td>
</tr>
<tr>
<td>VAL</td>
<td>55, 142, 151, 160, 162, 165, 172, 175, 200, 206, 221</td>
</tr>
<tr>
<td>BKCH</td>
<td>72, 110</td>
</tr>
<tr>
<td>CHAR</td>
<td>103</td>
</tr>
<tr>
<td>EOB</td>
<td>101</td>
</tr>
<tr>
<td>EVAL</td>
<td>23, 220</td>
</tr>
<tr>
<td>EVX4</td>
<td>52, 227</td>
</tr>
<tr>
<td>LPAR</td>
<td>132, 141</td>
</tr>
<tr>
<td>RCOMP</td>
<td>127</td>
</tr>
<tr>
<td>REX1</td>
<td>234</td>
</tr>
<tr>
<td>REX2</td>
<td>235</td>
</tr>
<tr>
<td>RPAR</td>
<td>135, 241</td>
</tr>
<tr>
<td>SAVE</td>
<td>216</td>
</tr>
<tr>
<td>SCAN</td>
<td>104</td>
</tr>
<tr>
<td>TCOM</td>
<td>14, 35, 40, 231</td>
</tr>
<tr>
<td>TEDF</td>
<td>13, 37, 100, 105, 232, 237</td>
</tr>
<tr>
<td>TERM</td>
<td>54, 173, 176, 201, 202, 211, 213, 226</td>
</tr>
<tr>
<td>BLANK</td>
<td>124</td>
</tr>
<tr>
<td>CRPAR</td>
<td>136</td>
</tr>
<tr>
<td>GEVAL</td>
<td>51</td>
</tr>
<tr>
<td>LBKCH</td>
<td>60, 170, 205</td>
</tr>
<tr>
<td>NLDCP</td>
<td>166</td>
</tr>
<tr>
<td>NRPAR</td>
<td>106, 136</td>
</tr>
<tr>
<td>RBKCH</td>
<td>136, 204</td>
</tr>
<tr>
<td>RSCN</td>
<td>61, 214, 215, 224</td>
</tr>
<tr>
<td>TABBC</td>
<td>70, 111, 136, 171, 207</td>
</tr>
<tr>
<td>RVREVL</td>
<td>150</td>
</tr>
<tr>
<td>UNSAVE</td>
<td>222</td>
</tr>
<tr>
<td>VAREVL</td>
<td>5</td>
</tr>
</tbody>
</table>

NO ERROR IN ABOVE ASSEMBLY.
TIME SPENT IN FAP... 000010 IN HUNDREDTHS OF MINUTES.
<table>
<thead>
<tr>
<th>PCC</th>
<th>COUNT</th>
<th>70</th>
<th>BINARY CARD LABEL.</th>
<th>OPTB0010</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>OPTBL</td>
<td>ENTRY</td>
<td>OPTB0020</td>
</tr>
<tr>
<td>00002</td>
<td>ENTRY</td>
<td>OPTBL</td>
<td>ENTRY TO POINTER WORD.</td>
<td>OPTB0040</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OPTB0050</td>
</tr>
</tbody>
</table>

**LINKAGE DIRECTOR**

<table>
<thead>
<tr>
<th>00002</th>
<th>00104</th>
<th>00003</th>
<th>OPTBL</th>
<th>PZE</th>
<th>*+1,0,LTH</th>
<th>CONTROL WORD.</th>
<th>OPTB0060</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OPTB0070</td>
</tr>
<tr>
<td>00003</td>
<td>00002213243</td>
<td>BCI</td>
<td>1,000AACL</td>
<td>CAP MNEMONIC.</td>
<td>OPTB0080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00004</td>
<td>+036100000000</td>
<td>OCT</td>
<td>036100000000</td>
<td>7090 INSTRUCTION.</td>
<td>OPTB0090</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00005</td>
<td>00000214521</td>
<td>BCI</td>
<td>1,C2CANA</td>
<td>OPTB0100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00006</td>
<td>-032000000000</td>
<td>OCT</td>
<td>432000000000</td>
<td>OPTB0110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00007</td>
<td>00000232143</td>
<td>BCI</td>
<td>1,000CAL</td>
<td>OPTB0120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00010</td>
<td>-050000000000</td>
<td>OCT</td>
<td>450000000000</td>
<td>OPTB0130</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00011</td>
<td>00000233062</td>
<td>BCI</td>
<td>1,C00CHS</td>
<td>OPTB0140</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00012</td>
<td>+076000000002</td>
<td>OCT</td>
<td>076000000002</td>
<td>OPTB0150</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00013</td>
<td>00000234321</td>
<td>BCI</td>
<td>1,000CLLA</td>
<td>OPTB0160</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00014</td>
<td>+050000000000</td>
<td>OCT</td>
<td>050000000000</td>
<td>OPTB0170</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00015</td>
<td>00000234362</td>
<td>BCI</td>
<td>1,000CLS</td>
<td>OPTB0180</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00016</td>
<td>+050200000000</td>
<td>OCT</td>
<td>050200000000</td>
<td>OPTB0190</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00017</td>
<td>00000234644</td>
<td>BCI</td>
<td>1,000COM</td>
<td>OPTB0200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00020</td>
<td>+076000000006</td>
<td>OCT</td>
<td>076000000006</td>
<td>OPTB0210</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00021</td>
<td>00000261224</td>
<td>BCI</td>
<td>1,C0CFAD</td>
<td>OPTB0220</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00022</td>
<td>+030000000000</td>
<td>OCT</td>
<td>030000000000</td>
<td>OPTB0230</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00023</td>
<td>00000262447</td>
<td>BCI</td>
<td>1,000DFDP</td>
<td>OPTB0240</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00024</td>
<td>+021000000000</td>
<td>OCT</td>
<td>021000000000</td>
<td>OPTB0250</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00025</td>
<td>00000264447</td>
<td>BCI</td>
<td>1,C0FMP</td>
<td>OPTB0260</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00026</td>
<td>+026000000000</td>
<td>OCT</td>
<td>026000000000</td>
<td>OPTB0270</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00027</td>
<td>00000266222</td>
<td>BCI</td>
<td>1,000FSB</td>
<td>OPTB0280</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00030</td>
<td>+030200000000</td>
<td>OCT</td>
<td>030200000000</td>
<td>OPTB0290</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00031</td>
<td>00000432123</td>
<td>BCI</td>
<td>1,G0LAC</td>
<td>OPTB0300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00032</td>
<td>+035300000000</td>
<td>OCT</td>
<td>035300000000</td>
<td>OPTB0310</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00033</td>
<td>00000432162</td>
<td>BCI</td>
<td>1,G0LAS</td>
<td>OPTB0320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00034</td>
<td>-034000000000</td>
<td>OCT</td>
<td>434000000000</td>
<td>OPTB0330</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00035</td>
<td>00000432263</td>
<td>BCI</td>
<td>1,G0LST</td>
<td>OPTB0340</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00036</td>
<td>+076000000001</td>
<td>OCT</td>
<td>076000000001</td>
<td>OPTB0350</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00037</td>
<td>00000432450</td>
<td>BCI</td>
<td>1,000LDQ</td>
<td>OPTB0360</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00040</td>
<td>+056000000000</td>
<td>OCT</td>
<td>056000000000</td>
<td>OPTB0370</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00041</td>
<td>00000432743</td>
<td>BCI</td>
<td>1,G0LGL</td>
<td>OPTB0380</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00042</td>
<td>-076300000000</td>
<td>OCT</td>
<td>476300000000</td>
<td>OPTB0390</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00043</td>
<td>00000432751</td>
<td>BCI</td>
<td>1,000LGR</td>
<td>OPTB0400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00044</td>
<td>-076500000000</td>
<td>OCT</td>
<td>476500000000</td>
<td>OPTB0410</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00045</td>
<td>00000436721</td>
<td>BCI</td>
<td>1,G0LKA</td>
<td>OPTB0420</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00046</td>
<td>+053400000000</td>
<td>OCT</td>
<td>053400000000</td>
<td>OPTB0430</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00047</td>
<td>00000465121</td>
<td>BCI</td>
<td>1,G0QRA</td>
<td>OPTB0440</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00050</td>
<td>-050100000000</td>
<td>OCT</td>
<td>450100000000</td>
<td>OPTB0450</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00051</td>
<td>00000472263</td>
<td>BCI</td>
<td>1,G0PB</td>
<td>OPTB0460</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00052</td>
<td>-076000000001</td>
<td>OCT</td>
<td>476000000001</td>
<td>OPTB0470</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00053</td>
<td>00000515043</td>
<td>BCI</td>
<td>1,G0QRL</td>
<td>OPTB0480</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00054</td>
<td>-077300000000</td>
<td>OCT</td>
<td>477300000000</td>
<td>OPTB0490</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00055</td>
<td>00000624366</td>
<td>BCI</td>
<td>1,G0SLW</td>
<td>OPTB0500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00056</td>
<td>+060200000000</td>
<td>OCT</td>
<td>060200000000</td>
<td>OPTB0510</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00057</td>
<td>00000626346</td>
<td>BCI</td>
<td>1,G0STO</td>
<td>OPTB0520</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OPERATION TABLE FOR CAP.

<table>
<thead>
<tr>
<th>OCT</th>
<th>BCI</th>
<th>060100000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>00060</td>
<td>+060100000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00061</td>
<td>000000000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00062</td>
<td>-060000000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00063</td>
<td>000000000626721</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00064</td>
<td>+0603400400000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00065</td>
<td>0000000033167</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00066</td>
<td>+0200014000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00067</td>
<td>00000000634431</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00068</td>
<td>-012000000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00069</td>
<td>00000000634743</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00070</td>
<td>+012000000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00071</td>
<td>000000006345121</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00072</td>
<td>+016200000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00073</td>
<td>00000000635047</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00074</td>
<td>+016200000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00075</td>
<td>00000000635121</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00076</td>
<td>+020000000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00077</td>
<td>00000000636267</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00078</td>
<td>+007400400000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00079</td>
<td>00000000637125</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00080</td>
<td>+010000000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00081</td>
<td>00000000672321</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00082</td>
<td>+013100000000</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00083</td>
<td>00000000672343</td>
<td>BCI 10000000</td>
</tr>
<tr>
<td>00084</td>
<td>-013000000000</td>
<td>BCI 10000000</td>
</tr>
</tbody>
</table>

00104  LTH EQU *-OPTBL-1  2*(NUMBER OF ALLOWED OPERATIONS).

END

POST PROCESSOR ASSEMBLY DATA

107 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS
104  LTH   2, 107
2  OPTBL  107

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP: 00.0004 IN HUNDREDTHS OF MINUTES.
INTOP, EVALUATE INT PSEUDO-OP.

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>OPERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>00002</td>
<td>0634 00 4 00112 INTPC SXA RX4,4 SAVE IRS.</td>
</tr>
<tr>
<td>00003</td>
<td>0634 00 2 00113 SXA RX2,2 **</td>
</tr>
<tr>
<td>00004</td>
<td>0634 00 1 00066 SXA HX1,1 ++</td>
</tr>
<tr>
<td>00005</td>
<td>-0500 00 4 00001 CAL 1,4 GET BUFFER ORIGIN.</td>
</tr>
<tr>
<td>00006</td>
<td>0361 00 0 00122 ACL =12 FORM BUFF+12.</td>
</tr>
<tr>
<td>00007</td>
<td>0621 00 0 00014 STA SCAN **</td>
</tr>
<tr>
<td>00010</td>
<td>0600 00 0 00115 STZ INT CLEAR CONVERSION.</td>
</tr>
<tr>
<td>00011</td>
<td>0600 00 0 00117 STZ TER RESET ERROR MARK.</td>
</tr>
<tr>
<td>00012</td>
<td>0600 00 0 00120 STZ TGQ RESET DIGIT MARK.</td>
</tr>
<tr>
<td>00013</td>
<td>0774 00 2 00012 AXT 10,2 SCAN TEN WORDS.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>OPERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>00014</td>
<td>0560 00 2 00000 SCAN LDQ **,2 GET BUFFER WORD. ADDRESS IS BUFF+12.</td>
</tr>
<tr>
<td>00015</td>
<td>0774 00 4 00006 AXT 6,4 SIX CHARACTERS.</td>
</tr>
<tr>
<td>00016</td>
<td>-0754 00 0 00000 PDQ 0,0 CLEAR AC.</td>
</tr>
<tr>
<td>00017</td>
<td>-0763 00 0 00006 LGL 6 GET CHARACTER.</td>
</tr>
<tr>
<td>00020</td>
<td>-0340 00 0 00121 LAS =1G CHECK FCR DIGIT.</td>
</tr>
<tr>
<td>00021</td>
<td>0020 00 0 00036 TRA CHAR MUST BE CHARACTER.</td>
</tr>
<tr>
<td>00022</td>
<td>0020 00 0 00054 TRA ERROR NO CHARACTER FOR CODE TEN.</td>
</tr>
<tr>
<td>00023</td>
<td>-0625 00 0 00120 STL TGQ DIGIT ENCOUNTERED, SET MARK.</td>
</tr>
<tr>
<td>00024</td>
<td>0600 00 0 00116 STG DIG SAVE DIGIT.</td>
</tr>
<tr>
<td>00025</td>
<td>0500 00 0 00115 CLA INT PROGRAMMED MULTIPLICATION OF INT BY TEN.</td>
</tr>
<tr>
<td>00026</td>
<td>0767 00 0 00002 ALS 2 *+INT.</td>
</tr>
<tr>
<td>00027</td>
<td>0400 00 0 00115 ADD INT 4**+INT=5**INT.</td>
</tr>
<tr>
<td>00030</td>
<td>0767 00 0 00001 ALS 1 2*4**+INT=10**INT.</td>
</tr>
<tr>
<td>00031</td>
<td>0361 00 0 00116 ACL DIG ADD DIGIT, IGNORING SIGN.</td>
</tr>
<tr>
<td>00032</td>
<td>0601 00 0 00115 STG INT SAVE.</td>
</tr>
<tr>
<td>00033</td>
<td>2 00001 4 00016 RSCAN TIX SCAN+2,4,1 COUNT CHARACTERS.</td>
</tr>
<tr>
<td>00034</td>
<td>2 00001 2 00014 TIX SCAN,2,1 COUNT WORDS.</td>
</tr>
<tr>
<td>00035</td>
<td>0020 00 0 00102 TRA BLANK END OF FIELD EQUIVALENT TO BLANK.</td>
</tr>
<tr>
<td>00036</td>
<td>0774 00 1 00010 CHAR AXT NBK,1 COMPARISON LOOP, GET NUMBER OF BREAKS.</td>
</tr>
<tr>
<td>00037</td>
<td>-0340 00 1 00054 LAS TABBK+NBK,1 COMPARE WITH TABLE.</td>
</tr>
<tr>
<td>00040</td>
<td>0020 00 0 00042 TRA *+2 NOT THIS ONE, TRY AGAIN.</td>
</tr>
<tr>
<td>00041</td>
<td>TRA TABBK+NBK+1,1 BREAKS, GO TO IT.</td>
</tr>
<tr>
<td>00042</td>
<td>2 0002 1 00037 TIX *+3,1,2 NOT THIS ONE, INDEX AND TRY AGAIN.</td>
</tr>
<tr>
<td>00043</td>
<td>0020 00 0 00054 TRA ERROR CANT FIND BREAK, ERROR.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>OPERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>00044</td>
<td>0000000000020 TABBK BCI 1,00000+ BREAK TABLE, PLUS.</td>
</tr>
<tr>
<td>00045</td>
<td>0020 00 0 00057 TRA PLUS **</td>
</tr>
<tr>
<td>00046</td>
<td>0000000000040 BCI 1,00000- MINUS.</td>
</tr>
<tr>
<td>00047</td>
<td>002C 00 0 00062 TRA MLAUS **</td>
</tr>
<tr>
<td>00050</td>
<td>0000000000073 BCI 1,00000, COMMA.</td>
</tr>
<tr>
<td>00051</td>
<td>0020 00 0 00066 TRA COMMA **</td>
</tr>
<tr>
<td>00052</td>
<td>0000000000060 BCI 1,00000 BLANK.</td>
</tr>
<tr>
<td>00053</td>
<td>0020 00 0 00102 TRA BLANK **</td>
</tr>
<tr>
<td>00010</td>
<td>NBK EQU --TABBK LENGTH OF BREAK TABLE.</td>
</tr>
</tbody>
</table>

99
INTOP, EVALUATE INT PSEUDO-OP.

MARK INTOP ERROR.
MARK ERROR IN THIS WORD.
RESUME SCAN.
INTOP0530
INTOP0540
INTOP0550
INTOP0560
INTOP0570
INTOP0580
INTOP0590
INTOP0600
INTOP0610
INTOP0620
INTOP0630
INTOP0640
INTOP0650
INTOP0660
INTOP0670
INTOP0680
INTOP0690
INTOP0700
INTOP0710
INTOP0720
INTOP0730
INTOP0740
INTOP0750
INTOP0760
INTOP0770
INTOP0780
INTOP0790
INTOP0800
INTOP0810
INTOP0820
INTOP0830
INTOP0840
INTOP0850
INTOP0860
INTOP0870
INTOP0880
INTOP0890
INTOP0900
INTOP0910
INTOP0920
INTOP0930
INTOP0940
INTOP0950
INTOP0960
INTOP0970
INTOP0980
INTOP0990

LITERALS
000000 000000 000012
000000 000000 000014
INTOP, EVALUATE INT PSEUDO-OP.
POST PROCESSOR ASSEMBLY DATA

123 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

116  DIG  24, 31
66  HX1  4, 100, 102
115  INT  10, 25, 27, 32, 64, 67, 75, 103
10  NBK  36, 37, 41, 54
113  RX2  3
112  RX4  2
120  TCG  12, 23, 57, 62, 76
117  TER  11, 55, 71, 73, 105, 107
36  CHAR  21
57  PLUS  45
14  SCAN  7, 33, 34
102  BLANK  35, 53
66  COPMA  51, 102
54  ERRCR  22, 43, 60, 63
2  INICP
62  MINUS  47
33  RSCAN  56, 61, 65, 101
44  TABBK  37, 41, 54

NO ERROR IN ABCVF ASSEMBLY.
*TIME SPENT IN FAP..  000005 IN HUNDREDTHS OF MINUTES.
UTILITY PROGRAMS FOR CAP.

PCC  UTIL0010
COUNT 114  UTIL0020  BINARY CARD LABEL.
LRL UTIL0040
ENTRY COMMA UTIL0050  COUNT COMMAS IN VARIABLE FIELD.
00024 ENTRY SAVE UTIL0060  ENTER WORDS IN PUSH-DOWN LIST.
00052 ENTRY UNSAVE UTIL0070  REMOVE WORDS FROM PUSH-DOWN LIST.
00074 ENTRY SCAN UTIL0080  ONE WORD EDITOR.
00092

$SCAN, REMOVE BLANKS FROM THE
WORD IN LAC, AND COMPRESS TO RIGHT.
UTIL0100  UTIL0110
UTIL0120

LINKAGE DIRECTOR
C0000 000000000000
C0001 234644442160

00002 0634 00 0 00021 SCAN SXA SCX4,4  SAVE IR4.  UTIL0130
C0003 -0130 00 0 00000 XCL  UTIL0140  PLACE WORD IN MQ.
00004 0774 00 0 00006 XAT  UTIL0150  COUNT SIX CHARACTERS.
00005 0660 00 0 00023 STZ  UTIL0160  WORD CLEAR COMPRESSED WORD.
00006 -0754 00 0 00000 SLOOP PXD  UTIL0170  0,0 CLEAR AC.
00007 -0763 00 0 00006 LGL  UTIL0180  6 GET CHARACTER.
00010 -0340 00 0 01107 LAS  UTIL0190  =060 CHECK FOR BLANK.
00011 0020 00 0 00013 TRA  UTIL0200  **2 NO, SKIP.
00012 0020 00 0 00017 TRA  UTIL0210  RSCAN YES, IGNORE.
00013 -0765 00 0 00006 LGR  UTIL0220  6 NO BLANK, BUILD WORD.
00014 -0500 00 0 00023 CAL  UTIL0230  WORD **
00015 -0763 00 0 00006 LGL  UTIL0240  6 **
00016 0602 00 0 00023 SLW  UTIL0250  WORD SAVE PARTIAL WORD.
00017 2 00001 4 00006 RSCAN TIX  UTIL0260  SLOOP,4,1 COUNT CHARACTERS.
00020 -0500 00 0 00023 CAL  UTIL0270  WORD GET COMPRESSED WORD.
00021 0774 00 4 00000 SCX  UTIL0280  **,4 RESTORE IR4.
00024 AXT  UTIL0290  **,4 RETURN WITH RESULT IN AC.
00022 0020 00 4 00001 TRA  UTIL0300  1,4
00023 0 00000 0 00000 WORD PZE  UTIL0310  STORAGE FOR PARTIAL WORD.
00026

END OF SCAN.
UTILITY PROGRAMS FOR CAP.

EJECT

$COMMA, COUNT COMMAS IN VARIABLE
FIELD PLUS ONE TO FIRST BLANK OR COLUMN
72, COUNT IS SUBTRACTED FROM IRI.

00024 0634 00 4 00047 COMMA SXA CDX4,4 SAVE IRS.
00025 0634 00 2 00050 SXA CDX2,2 " "
00026 -0500 00 4 00001 CAL 1,4 GET BUFFER ADDRESS.
00027 0361 00 0 01106 ACL *12 PLUS 12.
00030 0621 00 0 00033 STA LDG STA IN PICKUP.
00031 0774 00 4 00012 AXT 10,4 SCAN BUFFER+2 TO BUFF+12.
00032 0774 00 2 00006 CLP4 AXT 6,2 SIX CHARACTERS.
00033 0560 00 4 00000 LDQ LDQ **,4 GET WORD.
00034 -0754 00 0 00000 CLP2 PXO 0,0 CLEAR AC.
00035 -0763 00 0 00006 LGL 6 GET CHARACTER.
00036 -0340 00 0 01110 LAS =HC0000, CHECK FOR COMMA.
00037 0020 00 0 00044 TRA RCLP NO, CANT BE BLANK.
00040 1 77777 1 00044 TXI RCLP,1,-1 YES, COUNT AND RESUME SCAN.
00041 -0340 00 0 01107 LAS =HC0000 CHECK FOR BLANK.
00042 0020 00 0 00044 TRA **2 NO, SKIP.
00043 0020 00 0 00046 TRA ECSCN END OF COMMA SCAN.
00044 2 00001 2 00034 RCLP TIX CLP2,2,1 COUNT CHARACTERS.
00045 2 00001 4 00032 TIX CLP4,4,1 COUNT WORDS.
00046 1 77777 1 00047 ECSCN TXI **1,1,-1 COUNT LAST BLANK OR E.O.F.
00047 0774 00 4 00000 CDX4 AXT **,4 RESTORE IRS.
00050 0774 00 2 00000 CDX2 AXT **,2 " "
00051 0020 00 4 00002 TRA 2,4 RETURN.

END OF COMMA.
### Utility Programs for CAP

#### Eject

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Ate</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00052</td>
<td>0634</td>
<td>00</td>
<td>SAVE SXA</td>
<td>** SAVE IRS.</td>
</tr>
<tr>
<td>00053</td>
<td>0634</td>
<td>00</td>
<td>SXA SVX2,2</td>
<td>** GET CONTROL WORD.</td>
</tr>
<tr>
<td>00054</td>
<td>-0500</td>
<td>00</td>
<td>CAL 1,4</td>
<td>** COUNT TC IR2.</td>
</tr>
<tr>
<td>00055</td>
<td>-0734</td>
<td>00</td>
<td>PDX 0,2</td>
<td>** COUNT TC A(I).</td>
</tr>
<tr>
<td>00056</td>
<td>0754</td>
<td>00</td>
<td>PXA 0,2</td>
<td>** ADDRESS OF FIRST) + COUNT.</td>
</tr>
<tr>
<td>00057</td>
<td>0361</td>
<td>00</td>
<td>ACL 1,4</td>
<td>** STA IN PIPUP.</td>
</tr>
<tr>
<td>00058</td>
<td>0062</td>
<td>00</td>
<td>STA ++2</td>
<td>** CURRENT STORAGE COUNT TO IR4.</td>
</tr>
<tr>
<td>00059</td>
<td>0774</td>
<td>00</td>
<td>SCNT XTX</td>
<td>** **2 GET WORD. ** ** BCS OF CURRENT BLOCK</td>
</tr>
<tr>
<td>00060</td>
<td>0200</td>
<td>00</td>
<td>CAL SVX4</td>
<td>** PLACE IN LIST.</td>
</tr>
<tr>
<td>00061</td>
<td>0602</td>
<td>00</td>
<td>SLW SBBF+SVN4</td>
<td>** LIST COUNT.</td>
</tr>
<tr>
<td>00062</td>
<td>0500</td>
<td>00</td>
<td>TIX **3,4,1</td>
<td>** LIST EXCEEDED, SET INDICATOR,</td>
</tr>
<tr>
<td>00063</td>
<td>0602</td>
<td>00</td>
<td>TIX **5,2,1</td>
<td>** AND EXIT.</td>
</tr>
<tr>
<td>00064</td>
<td>0200</td>
<td>00</td>
<td>SVA SVX4</td>
<td>** SAVE WORDS TRANSMITTED.</td>
</tr>
<tr>
<td>00065</td>
<td>0000</td>
<td>00</td>
<td>SVA SCNT4</td>
<td>** SAVE LIST COUNT.</td>
</tr>
<tr>
<td>00066</td>
<td>0774</td>
<td>00</td>
<td>SVA SX4</td>
<td>** RESTORE IRS.</td>
</tr>
<tr>
<td>00067</td>
<td>0774</td>
<td>00</td>
<td>SX2 ATM **2</td>
<td>** RETURN.</td>
</tr>
<tr>
<td>00068</td>
<td>0200</td>
<td>00</td>
<td>SVA ATM 2,4</td>
<td>** END.</td>
</tr>
</tbody>
</table>

#### Unsafe

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Ate</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00074</td>
<td>0634</td>
<td>00</td>
<td>UNSX4,4</td>
<td>** SAVE IRS.</td>
</tr>
<tr>
<td>00075</td>
<td>0634</td>
<td>00</td>
<td>SXA UNSX2,2</td>
<td>** GET CONTROL WORD.</td>
</tr>
<tr>
<td>00076</td>
<td>-0500</td>
<td>00</td>
<td>CAL 1,4</td>
<td>** INSERT COUNT.</td>
</tr>
<tr>
<td>00077</td>
<td>0622</td>
<td>00</td>
<td>STD 18</td>
<td>** COUNT TC A(I).</td>
</tr>
<tr>
<td>01000</td>
<td>0771</td>
<td>00</td>
<td>ARS 1,4</td>
<td>** ADDRESS OF FIRST) + COUNT.</td>
</tr>
<tr>
<td>01001</td>
<td>0361</td>
<td>00</td>
<td>ALO 0,2</td>
<td>** STA IN STORE.</td>
</tr>
<tr>
<td>01002</td>
<td>0621</td>
<td>00</td>
<td>AX 1,2</td>
<td>** SETUP FOR WORD COUNT.</td>
</tr>
<tr>
<td>01003</td>
<td>0774</td>
<td>00</td>
<td>LXA SCNT4</td>
<td>** LIST COUNT TO IR4.</td>
</tr>
<tr>
<td>01004</td>
<td>0534</td>
<td>00</td>
<td>LXA ATM **3,4,SVN</td>
<td>** IS LIST EXCEEDED.</td>
</tr>
<tr>
<td>01005</td>
<td>1000</td>
<td>00</td>
<td>LXA ATM 4</td>
<td>** YES, SET INDICATOR,</td>
</tr>
<tr>
<td>01006</td>
<td>-3007</td>
<td>64</td>
<td>LXA ATM 4</td>
<td>** AND EXIT.</td>
</tr>
<tr>
<td>01007</td>
<td>-0555</td>
<td>00</td>
<td>LXA ATM 4</td>
<td>** OK, GET WORD.</td>
</tr>
<tr>
<td>01011</td>
<td>0020</td>
<td>00</td>
<td>TIX SBBF+SVN4</td>
<td>** CLEAR LIST.</td>
</tr>
<tr>
<td>01012</td>
<td>-0500</td>
<td>00</td>
<td>TIX SBBF+SVN4</td>
<td>** INSERT IN CALLING PROGRAM.</td>
</tr>
<tr>
<td>01013</td>
<td>0602</td>
<td>00</td>
<td>TIX SBBF+SVN4</td>
<td>** COMPARE WITH BLOCK LENGTH.</td>
</tr>
<tr>
<td>01014</td>
<td>1000</td>
<td>00</td>
<td>TIX SBBF+SVN4</td>
<td>** SAVE LIST COUNT.</td>
</tr>
<tr>
<td>01015</td>
<td>-3000</td>
<td>00</td>
<td>TIX SBBF+SVN4</td>
<td>** RESTORE IRS.</td>
</tr>
<tr>
<td>01016</td>
<td>0634</td>
<td>00</td>
<td>TIX SBBF+SVN4</td>
<td>** RETURN.</td>
</tr>
</tbody>
</table>

#### Literals

- 01106: 000000000001
- 01107: 000000000060
- 01110: 000000000073
UTILITY PROGRAMS FOR CAP.
POST PROCESSOR ASSEMBLY DATA

1111 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

33  LCQ  30
764  SVN  61,  63,  106,  111,  112,  122,  1106
34  CLP2  44
32  CLP4  45
50  CDX2  25
47  CDX4  24
44  RCLP  37,  40
52  SAVE
2  SCAN
61  SCNT  70,  104,  116
21  SCX4  2
72  SVX2  53
71  SVX4  52,  66
23  WORD  5,  14,  16,  20
24  COMMA
46  ESSCN  43
17  RSCAN  12
122  SBUF  63,  111,  112
6  SLOCP  17
105  UNSLP
113  UNSLH  102,  115
120  UNSX2  75
117  UNSX4  74,  110
115  UNTXL  77
74  UNSAVE

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP..  000006 IN HUNDREDTHS OF MINUTES.
$SYMSTO, AND $SYMGET, OPERATIONS WITH SYMBOL TABLE.

PCC
COUNT  54

LBL
ENTRY $SYMSTO  BINARY CARD LABEL.
ENTRY $SYMGET  ENTRY TO LOOK-UP VALUE OF SYMBOL.
ENTRY $PSMTBL  POINTER TO SYMBOL TABLE AND SIZE.
ENTRY $SYMSTO  ENTRY TO PLACE SYMBOL AND VALUE IN TABLE.

$SYMSTO, FORM SYMBOL TABLE.

TRANSFER VECTOR
00000  622321456060  SCAN

LINKAGE DIRECTOR
00001  000000000000
00002  627044272563

00003 -0340 00 0 00353  $SYMSTO LAS  *H  CHECK FOR BLANK LOCATION FIELD.
00004  0020 00 0 00006  TRA  **2  NOT BLANK, SKIP.
00005  0020 00 4 00011  TRA  1,4  BLANK, DON'T STORE, RETURN TO CALLER.
00006  0634 00 4 00024  SXA  SSX4,4  SAVE IR4.
00007  0074 00 4 00000  TSX  *SCAN,4  COMpress SYMBOL TO RIGHT.
00010 -0130 00 0 00000  XCL  PLACE SYMBOL IN MQ.
00011  0754 00 1 00000  PXA  0,1  GET *TILC1.
00012  0737 00 4 00000  PAC  0,4  **
00013  0754 00 4 00000  PXA  0,4  **
00014 -0534 00 4 00042  LXD  PSMTBL,4  GET CURRENT COUNT OF TABLE.
00015  100003 4 00016  TXI  **1,4,2  MAKE ROOM FOR ONE MORE.
00016 -3 00310 4 00021  TXL  **3,4,LSMTBL  CHECK FOR TABLE OVERFLOW.
00017  0055 00 0 00001  S1L  1  SYMBl EXCEEDED, SET INDICATOR.
00020  0020 00 0 00024  TRA  SSX4  GO TO RETURN.
00021 -0600 00 4 00353  STQ  SYMTBL,4  SAVE SYMBOL.
00022  0602 00 4 00354  SWX  SYMTBL+1,4  SAVE VALUE.
00023 -0634 00 4 00042  SXD  PSMTBL,4  SAVE TABLE COUNT.
00024  0774 00 4 00000  SSX4  AXT  **,4  RESTORE IR4.
00025  0020 00 4 00001  TRA  1,4  RETURN.

SPACE 2

$SYMGET, LOOK UP SYMBOL AND GET VALUE.

00026  0634 00 4 00040  $SYMGET SXA  SGX4,4  SAVE IR4.
00027 -0534 00 4 00042  LXD  PSMTBL,4  GET TABLE COUNT.
00030 -0340 00 4 00333  LAS  SYMTBL,4  COMPARE WITH TABLE.
00031  0020 00 0 00033  TRA  **2  NOT THIS ONE, SKIP.
00032  0020 00 0 00037  STQ  SYMFND  FOUND, EXIT.
00033  2 00002 4 0030  TIX  **3,4,2  INDEX AND TRY AGAIN.
00034 -0754 00 0 00000  PXD  0,0  NOT FOUND, VALUE IS ZERO.
00035  0055 00 0 00001  SIR  1  SET UNDEFINED SYMBOL INDICATOR.
00036  0020 00 4 00040  TRA  SGX4  GO TO EXIT.
00037 -0500 00 4 00354  SYMFND CAL  SYMTBL+1,4  FOUND, GET VALUE.
00040  0774 00 4 00000  SGX4  AXT  **,4  RESTORE IR4.
00041  0020 00 4 00001  TRA  1,4  RETURN.
$SYMSTO, AND $SYMGET, OPERATIONS WITH SYMBOL TABLE.

SPACE

2

STORAGE AND CONSTANTS.

SYMS0470
SYMS0480
SYMS0490
SYMS0500
SYMS0510
SYMS0520
SYMS0530
SYMS0540

00310  LSMTBL EQU 2*100
00042  0 00000 0 00353
00353  PSMTBL PZE SYMTBL,Q,** POINTER WORD TO SYMTBL.
6

SYMTBL BE$ LSMTBL SYMBOL TABLE.

END

LITERALS

00353 606060606060

POST PROCESSOR ASSEMBLY DATA

354 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

0 SCAM 7
40 SGX4 26, 36
24 SSX4 6, 20
310 LSMTBL 16, 42, 353
42 PSMTBL 14, 23, 27
37 SYMFND 32
26 SYMGET
3 SYMSTO
353 SYMTBL 21, 22, 30, 37, 42

NO ERROR IN ABOVE ASSEMBLY.
TIME SPENT IN FAP.. 000004 IN HUNDREDTHS OF MINUTES.
ENDP AND OTHER SUBROUTINES USED BY COMP.

<table>
<thead>
<tr>
<th>PCC</th>
<th>COUNT</th>
<th>LBL</th>
<th>ENTRY</th>
<th>GENOP</th>
<th>ENTRY</th>
<th>ILC</th>
<th>ENTRY</th>
<th>GNSTO</th>
<th>ENTRY</th>
<th>NSTO</th>
<th>ENTRY</th>
<th>ENDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>102</td>
<td>ENDP</td>
<td>PIVAR</td>
<td>GENOP</td>
<td>GENOP</td>
<td>ILG</td>
<td>GENOP</td>
<td>NSTO</td>
<td>ENDP</td>
<td>ENDP</td>
<td>ENDP</td>
<td></td>
</tr>
<tr>
<td>00036</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0040</td>
</tr>
<tr>
<td>00045</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0050</td>
</tr>
<tr>
<td>00053</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0060</td>
</tr>
<tr>
<td>00071</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0070</td>
</tr>
<tr>
<td>00080</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0080</td>
</tr>
<tr>
<td>00090</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0090</td>
</tr>
<tr>
<td>00100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0100</td>
</tr>
<tr>
<td>00110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0110</td>
</tr>
<tr>
<td>00120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0120</td>
</tr>
<tr>
<td>00130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0130</td>
</tr>
<tr>
<td>00140</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0140</td>
</tr>
</tbody>
</table>

ENTRY TO RESERVE STORAGE.

TRANSFER VECTOR

00000 627044626346 SYMSTO
00001 662363016060 WCTI

LINKAGE DIRECTR

00002 000000000000
00003 473165215160

ENDP NIT MSTO IF NO STORAGE ALLOCATED,

0004 -0520 00 00 00123 TRA 1,4 RETURN TO CALLER.
0005 0020 00 4 00001 SXA ENDX4,4 STORAGE ALLOCATED, SAVE IR4.
0006 0634 00 4 00016 CAL =HTEM INSERT THIS SYMBOL IN SYMBOL TABLE.
0007 -0500 00 00150 TSX $SYMSTO,4 **
0010 0074 00 4 00000 TSX $WCTI,4 ** PUT REM CARD ON CTL.
0011 0074 00 4 00001 TSX $WCTI,4 **
0012 0 0000 0 00020 PZE EBBUF **
0013 0535 00 4 00123 LAC MSTO,4 INCREASE ILC FOR STORAGE.
0014 -0634 00 4 00015 SXO **+1,4 **
0015 1 00000 1 00016 TXI **+1,1,**
0016 0774 00 4 00000 ENDX4 AXI **+4 RESTORE IR4.
0017 0020 00 4 00001 TRA 1,4 RETURN.

0020 606325446060 EBBUF BCI 9, TEM REM TEMPORARY STORAGE AREA BEGINS HERE.
0021 605125446660
0022 63254474681
0023 21517066263
0024 465121272560
0025 21512521622
0026 25273145620
0027 302551253360
0030 606060606060 BCI 5,
0031 606060606060
0032 606060606060
0033 606060606060
0034 606060606060
0035 606060606060

ENTRY TO FORM VARIABLE FIELD.

0036 0634 00 4 00043 PIVAR SXA PX4,4 SAVE IR4.
0037 0774 00 4 00013 PCNT AXI 11,4 COUNT 10 WORDS WITH TNX.
0040 -2 00001 4 00043 TNX PX4,4,1 INDEX WORD COUNT.
0041 0602 00 4 00141 SLW PBUFF+12,4 PLACE WORD IN BUFFER.
ENDOP AND OTHER SUBROUTINES USED BY COMP.

00042 0634 00 4 00037  SXA PX4 PCNT,4 SAVE WORD COUNT.
00043 0774 00 4 00000  AXT **,4 RESTORE IR4.
00044 0020 00 4 00001  TRA 1,4 RETURN.

SPACE 2
ENTRY TO INSERT OP-FIELD AND WGT1.

00045 0634 00 4 00056  GENCP SXA GOPX4,4 SAVE IR4.
00046 -0500 00 4 00001  CAL 1,4 GET OP.
00047 0602 00 0 00126  SLW PUFF,4 INSERT OP-FIELD.
00048 0074 00 4 00000  AXT $GT1,4 WRITE COLLATION TAPE.
00049 00000 0 00125  PZE PUFF ..
00050 0074 00 4 00000  TXS ERASE,4 CLEAR PUFF.
00051 0774 00 4 00000  ILC AXT **,4 INCREMENT ILC.
00052 0774 00 4 00000  SXA ILC,4 SAVE CURRENT ILC.
00053 0774 00 4 00000  GOPX4 AXT **,4 RESTORE IR4.
00054 0200 00 4 00022  TRA 2,4 RETURN.

SPACE 2
ENTRY TO ERASE PUFF.

00055 0634 00 4 00067  ERASE SXA ERX4,4 SAVE IR4.
00056 0774 00 4 00013  AXT 11,4 RESET PCNT.
00057 0634 00 4 00037  SXA PCNT,4 ..
00058 0774 00 4 00016  AXT 14,4 LOAD BUFFER WITH BLANKS.
00059 -0500 00 0 000145 CAL =H ..
00060 0602 00 4 00143  SLW PBUFF,14,4 ..
00061 2 00001 4 00065  TI1 =-1,4,1 ..
00062 0774 00 4 00000  EXR4 AXT **,4 RESTORE IR4.
00063 0020 00 4 00001  TRA 1,4 RETURN.

SPACE 2
ENTRY TO GET NEXT TEMPORARY STORAGE SYMBOL.

00064 0500 00 0 00122  GNSCL CLA NSTO PLACE NUMBER OF LAST STORAGE.
00065 0560 00 0 00122  LDQ NSTO IN AC AND MQ.
00066 0400 00 0 00143  ADD =1 INCREMENT AND SAVE FOR NEXT.
00067 0601 00 0 00122  STA NSTO ..
00068 034C 00 0 00123  CAS MSTD CHECK FOR MSTD EXCEEDED.
00069 0601 00 0 00123  STO MSTD YES; UPDATE MSTD.
00070 0761 00 0 00000  NOP EQUAL, IGNORE.
00071 0131 00 0 00000  XCA PLACE NSTO IN AC.
00072 -0100 00 0 00004  TNL **3 CHECK FOR ZERO NSTO.
00073 -0500 00 0 00150  CAL =HTEM ZERO, PICKUP CHARACTERS.
00074 0020 00 4 00001  TRA 1,4 RETURN TO CALLER.
00075 0340 00 0 00144  CAS =10 CHECK FOR ONLY ONE DIGIT.
00076 0020 00 0 00112  TRA TWDG TWO DIGITS.
00077 0020 00 0 00112  TRA TWDG ..
00078 0767 00 0 00006  ALS 6 ONE DIGIT, SHIFT TO POSITION.
00079 -0501 00 0 00147  QRA =HTEM+O INSERT CHARACTERS.
00080 0020 00 4 00001  TRA 1,4 RETURN TO CALLER.
00081 0131 00 0 00000  TWDG XCA TWO DIGITS, PLACE NSTO IN MQ AGAIN.
ENDP AND OTHER SUBROUTINES USED BY COMP.

00113 -0754 00 0 00000 XD 0, C CLEAR AC.
00114 0221 00 0 00144 DVP =10 MOD 10.
00115 -0773 00 0 00006 RQL 6 FIRST DIGIT TO K5.
00116 -0600 00 0 00124 STQ DNSTO SAVE.
00117 -0501 00 0 00124 ORA DNSTO FORM DEcimal STORAGE NUMBER.
00120 -0501 00 0 00146 ORA =HTEM+00 INSERT CHARACTERS.
00121 0020 00 4 00001 TRA 1,4 RETURN WITH RESULT IN AC.

SPACE 2
STORACE AND CONSTANTS.

00122 0 00000 0 00000 NSTC PZE CURRENT STORAGE COUNTER.
00123 0 00000 0 00000 MSTD PZE MAXIMUM STORAGE COUNTER.
00124 0 00000 0 00000 CNSTG PZE DECIMAL STORAGE COUNTER.
00125 PBUFF 154 STATEMENT BUFFER.

END

LITERALS
00143 000000000001
00144 000000000012
00145 60606060606060
00146 632542000000
00147 632542000060
00190 632544606060

POST PROCESSOR ASSEMBLY DATA

151 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS
53 ILC 55
43 PX4 36, 40
67 ERX4 60
123 MSTD 4, 13, 75, 76
122 NSTO 71, 72, 74
37 PCNT 42, 62
1 WC1 11, 50
124 DNSTO 116, 117
20 EBUFF 12
4 ENDP
16 ENDX4 6
60 ERASE 52
45 GENOP
71 GNSSTO
56 GOPX4 45
125 PBUFF 41, 47, 51, 65
36 PIVAR
112 TNDDG 105, 106
0 SYMSTO 10

NO ERROR IN ABOVE ASSEMBLY.
TIME SPENT IN FAP 000005 IN HUNDREDTHS OF MINUTES.
SUBROUTINE COMPOP, COMPILE ARITHMETICS FOR CAP.

PCC
COUNT 186
LBL COMPPOP BINARY CARD LABEL.
ENTRY COMPPOP EVALUATE 'COMP' PSEUDO-CP.

$COMPPOP IS CALLED BY,

TSX $COMPPOP,4
PZE BLFF
WHERE BUFF IS A 14 WORD BUFFER CONTAINING THE
HOLLERITH CARD IMAGE OF THE COMP STATEMENT.
COMPPOP TAKES THE VARIABLE FIELD AS A FORTRAN
ARITHMETIC STATEMENT AND COMPILES IN FLOATING
POINT. COMMAS ARE TREATED AS PART OF THE SYMBOL,
HENCE TAGGING IS ALLOWED. BLANKS ARE IGNORED.
COMPPOP OPERATES IN 2 PASSES. PASS1 TAKES THE
CARD IMAGE APART SEPARATING SYMBOLS FROM OPERATION
CHARACTERS. PASS2 EVALUATES PROS FORMS FROM THE
INNERMOST (...) PAIR OUTWARD. SOME OPTIMIZATION
IS DONE BUT THERE ARE NO DIAGNOSTICS.

TRANSFER VECTOR
00000 314323600600 ILC
00001 456263460000 NSIT
00002 255121622560 ERASE
00003 256747516060 EXPR
00004 473165215160 PIVAR
00005 272545464760 GENOP

LINKAGE DIRECTOR
00006 000000000000
00007 234644474647

00010 0634 00 4 00210 COMPPOP SXA RX4,4 SAVE IRS.
00011 0634 00 2 00211 SXA RX2,2 **
00012 0754 00 1 00000 PXA 0.1 GET -(ILC).
00013 0621 60 0 00000 STA* $ILC SAVE.
00014 -0500 00 4 00001 CAL 1,4 GET CONTROL WORD.
00015 0361 00 0 00536 ACL =12 BUFF=12.
00016 0210 00 0 00032 STA CALL STA IN PICKUP.
00017 0600 60 0 00001 STZ* &NSIT ZERO NUMBER OF TEMPORARY STORAGE.
00018 0600 00 0 00216 STZ TECF1 RESET EOF MARK.
00019 -0500 00 0 00534 CAL =1 SETUP SYM.
00020 0602 00 0 00215 SLW SYM **
00023 0074 00 4 00002 TSX $ERASE,+4 ERASE BUFFER.
00024 0774 00 4 00310 AXT LFLD,+4 SETUP FLD COUNT.
00025 0634 00 4 00124 SXA FCNT,+4 **
00026 0140 00 0 00027 TOV **1 TURN OFF OVERFLOW LIGHT.
00027 0020 00 0 00030 TRA CPAS1 GO TO PASS1.
SUBROUTINE COMPOP, COMPILE ARITHMETICS FOR CAP.

SPACE 2

PASS 1 OF COMP, SEPARATE FIELD INTO SYMBOL AND BREAKS.

00030 0774 00 2 00012 CPAS1 AXT 10,2 COUNT 10 WORDS IN VARIABLE FIELD.
00031 0774 00 1 00006 RSCNI AXT 6,1 COUNT 6 CHARACTERS.
00032 0500 00 2 00000 CAL CAL **,2 GET WORD.
00033 0340 00 0 00542 LAS =H CHECK FOR ALL.Blanks.
00034 0200 00 0 00036 TRA **2 NO, SKIP.
00035 0200 00 0 00563 TRA RSCNI+2 ALL BLANK, IGNORE.
00036 0130 00 0 00000 XCL PLACE IN MQ.
00037 0754 00 0 00000 SCNI PXD 0,0 CLEAR AC.
00040 0763 00 0 00006 LGL 6 GET NEXT CHARACTER.
00041 0600 00 0 00217 STQ MG SAVE MQ.
00042 0340 00 0 00540 LAS =H0000 CHECK FOR BLANK.
00043 0200 00 0 00000 TRA **2 NO, SKIP.
00044 0200 00 0 0061 TRA RSCNI YES, IGNORE.
00045 0774 00 4 00007 AXT NBK,4 CHECK FOR BREAK.
00046 0340 00 4 00075 LAS TABBK+NBK,4 COMPARE WITH TABLE.
00047 0200 00 0 0051 TRA **2 NO, SKIP.
00050 0200 00 0 0075 TRA BRK BREAK FOUND, EXIT.
00051 2 00001 4 00464 TIX -3,4,1 INDEX, AND TRY AGAIN.
00052 -0765 00 0 00006 LGR 6 BUILD SYM.
00053 0500 00 0 00215 CAL SYM **
00054 -0763 00 0 00006 LGL 6 **
00055 -0140 00 0 00000 TNO **3 IF SYMBOL FULL.
00056 0074 00 4 00123 TSX STFLD,4 INSERT IN LIST.
00057 0500 00 0 00534 CAL =1 AND BEGIN NEW SYMBOL.
00058 0602 00 0 00215 SLW SYM SAVE PARTIAL SYMBOL.
00059 0560 00 0 00217 RSCNI LOQ MG RESTORE MQ.
00060 0200 1 00001 2 0037 TIX SCNI+1,1 COUNT CHARACTERS.
00061 2 00001 1 00037 TIX BSCNI+1,1 COUNT WORDS.
00062 0200 1 00001 2 00316 STL TEOFI SET EOF MARK.
00063 0625 00 0 00216 TRA BRK GO PROCESS BREAK.
00064 0200 0 0 0075 TRA BRK GO PROCESS BREAK.
00065 0200 00 0 00000 TABBK BCI 1,0000+ TABLE OF BREAK CHARACTERS.
00066 00000000000200 TABBK BCI 1,0000- ** TABLE OF BREAK CHARACTERS.
00067 000000000040 BCI 1,0000- ** TABLE OF BREAK CHARACTERS.
00068 000000000054 BCI 1,0000- ** TABLE OF BREAK CHARACTERS.
00069 000000000061 BCI 1,0000- ** TABLE OF BREAK CHARACTERS.
00070 000000000077 BCI 1,0000+ ** TABLE OF BREAK CHARACTERS.
00071 000000000078 BCI 1,0000+ ** TABLE OF BREAK CHARACTERS.
00072 000000000074 BCI 1,0000+ ** TABLE OF BREAK CHARACTERS.
00073 000000000034 BCI 1,0000-= ** TABLE OF BREAK CHARACTERS.
00074 000000000013 BCI 1,0000-= ** TABLE OF BREAK CHARACTERS.
00075 0602 00 0 00220 BRK SLW LBRK SAVE BREAK CHARACTER.
00076 0500 00 0 00219 CAL SYM GET SYMBOL.
00077 -0340 00 0 00534 LAS =1 CHECK FOR NO CHARACTERS.
00100 0200 00 0 00102 TRA **2 YES, SKIP.
00101 0200 00 0 00110 TRA NOSYM NO, DONT STORE.
00102 0560 00 0 00542 LOQ 4** SYM, LEFT JUSTIFY.
00103 0763 00 0 00000 XCL 6 **
00104 -0140 00 0 00103 TNO 1 **
00105 0074 00 4 00123 TSX STFLD,4 PLACE IN FLD.
00106 0500 00 0 00534 CAL =1 BEGIN NEW SYMBOL.
00107 0602 00 0 00215 SLW SYM **
00110 0520 00 0 00216 NOSYM TEOFI CHECK FOR EOF.
00111 0020 0 0 00134 TRA EDFI EOF, GO TO IT. 

COMP0400
COMP0410
COMP0420
COMP0430
COMP0440
COMP0450
COMP0460
COMP0470
COMP0480
COMP0490
COMP0500
COMP0510
COMP0520
COMP0530
COMP0540
COMP0550
COMP0560
COMP0570
COMP0580
COMP0590
COMP0600
COMP0610
COMP0620
COMP0630
COMP0640
COMP0650
COMP0660
COMP0670
COMP0680
COMP0690
COMP0700
COMP0710
COMP0720
COMP0730
COMP0740
COMP0750
COMP0760
COMP0770
COMP0780
COMP0790
COMP0800
COMP0810
COMP0820
COMP0830
COMP0840
COMP0850
COMP0860
COMP0870
COMP0880
COMP0890
COMP0900
COMP0910
COMP0920
COMP0930
COMP0940
COMP0950
SUBROUTINE COMPOP, COMPILE ARITHMETICS FOR CAP.

00112 -0500 00 0 00220 CAL LBRK NOT EOF, GET BREAK CHARACTER.
00113 0074 00 4 00123 TSX STFLD,4 PLACE IN FIELD.
00114 -0500 00 0 00220 CAL LBRK GET BREAK CHARACTER.
00115 0322 00 0 00535 ERA **HOD000= CHECK FOR =.
00116 -0100 00 4 00061 TNZ RSCN1 NO, RESUME SCAN.
00117 -0500 00 0 00124 CAL FCNT YES, MARK TOP OF FIELD.
00118 0361 00 0 00534 ACL *1 **.
00119 0621 00 0 00221 STA TFLD **.
00120 0020 00 0 00061 TRA RSCN1 RESUME SCAN.
00121 0634 00 4 00130 STFLD SXA STX4,4 SAVE IR4.
00122 0774 00 4 00310 FCNT AXA LFLD,4 GET CURRENT FLD INDEX.
00123 0622 00 4 00534 SLW FLC,4 INSERT WORD.
00124 0012 00 4 00534 TNX XFLD,4,1 COUNT THIS WORD.
00125 0634 00 4 00124 SXA **3,4 SAVE COUNT.
00126 -0500 00 0 00000 STX4 AXA **4 RESTORE IR4.
00127 0774 00 4 00000 TRA 1,4 RETURN.
00128 -0550 00 0 00010 XFLD SIL 10 COMP STATEMENT TOO LONG, MARK ERROR.
00129 0200 00 0 00210 TRA RX4 AND RETURN.
00130 -0500 00 0 00124 EDF1 CAL FCNT MARK BOTTOM OF FIELD.
00131 0621 00 0 00222 STA BFLD **.
00132 0020 00 0 00137 TRA CPAS2 GO TO PASS2.

SPACE 2
PASSE 2 OF COMP, FNC AND EVALUATE EXPRESSIONS.

00137 0534 00 1 00222 CPAS2 LXA BFLD,1 SET CONTROL TXH.
00140 -0634 00 1 00155 SXD RSCN2+1,1 **.
00141 0534 00 1 00221 BSCN2 LXA TFLC,1 GET TOP OF FIELD.
00142 0634 00 1 00223 SXA LLLPAR,1 TREAT AS LAST (.).
00143 1 77777 1 00144 TX1 **1,1,1,-1 INDEX TO FIRST WORD.
00144 -0500 00 1 00534 SCN2 CAL FLC,1 GET NEXT FLD WORD.
00145 0100 00 0 00154 TZE RSCN2 IF ZERO, IGNORE.
00146 -0340 00 0 00541 LAS **HOD000 CHECK FOR .).
00147 0020 00 0 00151 TRA **2 NO, SKIP.
00148 0020 00 0 00157 TRA LPAR2 YES, GO TO IT.
00149 0340 00 0 00537 LAS **HOD00 CHECK FOR ).
00150 0020 00 0 00154 TRA **2 NO, SKIP.
00151 0020 00 0 00161 TRA RPAS2 YES, GO TO IT.
00152 1 77777 1 00153 RSCN2 TX1 **1,1,1,-1 COUNT WORDS.
00153 3 00000 1 00144 TXH SCN2,1,** CHECK FOR EOF.
00154 0020 00 0 00172 TRA EDF2 EOF, GO TO IT.

00157 0634 00 1 00223 LPAR2 SXA LLLPAR,1 SET LAST (.).
00160 0020 00 0 00154 TRA RSCN2 RESUME SCAN.
00161 -0634 00 1 00167 RPAS2 SXD EXPW1,1 EXPRESSION, MARK BOTTOM FOR $EXPR.
00162 0600 00 1 00534 STZ FLC,1 CLEAR FLD.
00163 0534 00 1 00223 LXA LLLPAR,1 GET INDEX OF LAST (.).
00164 0600 00 1 00534 STZ FLC,1 CLEAR FLD.
00165 0634 00 1 00167 SXA EXPW1,1 MARK TOP FOR $EXPR.
00166 0074 00 4 00003 TSX $EXPR,4 GO EVALUATE EXPRESSION.
00167 0 00000 0 00000 EXPW1 PZE **0,0,0,0 ZERO TAG MEANS STORE INTERMEDIATE.
SUBROUTINE COMPDPR, COMPIL ARITHMETICS FOR CAP.

C0170 0 00000 0 00534 PZE FLC   ***   COMP1500
C0171 0020 00 0 00141 TRA BSCN2   RESTART SCAN. COMP1510
C0172 -0634 00 1 00177 EDF2 SXD EXPW2,1   FINAL EXPR, MARK BOTTOM. COMP1530
C0173 0534 00 1 00223 LXA LLPAR,1   GET INDEX OF TOP. COMP1540
C0174 0600 00 1 00534 STZ FLC,1   CLEAR FLD. COMP1550
C0175 0634 00 1 00177 SXA EXPW2,1   MARK TOP. COMP1560
C0176 0074 00 4 00003 TSX $EXPR,4   EVALUATE EXPRESSION. COMP1570
C0177 0 00000 7 00000 EXPW2 PZE **,7,**   NON-ZER0 TAG MEANS LEAVE IN AC. COMP1580
C0200 0 00000 0 00534 PZE FLC   **   COMP1590
C0201 0774 00 1 00310 AXI LFLD,1   FORM FINAL STORAGE. COMP1600
C0202 -0500 00 1 00534 CAL FLD,1   GET WORD. COMP1610
C0203 0100 00 0 00206 TZE **3   IF ZERO, SYMBOL DONE. COMP1620
C0204 0074 00 4 00004 TSX $PIVAR,4   PLACE IN VARIABLE FIELD. COMP1630
C0205 1 7777 1 00202 TXI **3,1,-1   INDEX FOR NEXT WORD. COMP1640
C0206 0074 00 4 00005 TSX $GENOP,4   FINAL OP IS STO. COMP1650
C0207 606263466060 BCI 1, STO   **   COMP1660
C0210 0774 00 4 00000 RX4 AXI **,4   RESTORE IRS. COMP1670
C0211 0774 00 2 00000 RX2 AXI **,2   **   COMP1690
C0212 -0500 60 0 00000 CAL* $ILC   RESTORE ILC. COMP1700
C0213 0734 00 1 00000 PAX 0,1   **   COMP1710
C0214 0020 00 4 00002 TRA 2,4   RETURN. COMP1720

SPACE 2

STORAGE AND CONSTANTS.

C0215 0 00000 0 0000 SYM PZE   PARTIAL SYMBOL STORAGE. COMP1730
C0216 0 00000 0 0000 TEOF1 PZE   END OF FIELD MARK. COMP1740
C0217 0 00000 0 0000 PQ PZE   MQ STORAGE. COMP1750
C0220 0 00000 0 00000 LBK PZE   LAST BREAK. COMP1760
C0221 0 00000 0 00000 TFLD PZE   TOP OF FLD. COMP1770
C0222 0 00000 0 00000 BFLD PZE   BOTTOM OF FLD. COMP1780
C0223 0 00000 0 00000 LLPAR PZE   LAST I., COMP1790
C0230 00310 LFLD EQU 200   LENGTH OF FLD. COMP1800
C0534 FLD BES LFLD   ARITHMETIC FIELD BUFFER. COMP1810

LITERALS

00534 000000000001
00535 000000000013
00536 000000000014
00537 000000000034
00540 000000000060
00541 000000000074
00542 606060606060
SUBROUTINE COMPCP, COMPILE ARITHMETICS FOR CAP.
POST PROCESSOR ASSEMBLY DATA

543 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQ</td>
<td>41, 61</td>
</tr>
<tr>
<td>BRK</td>
<td>50, 65</td>
</tr>
<tr>
<td>FLD</td>
<td>125, 144, 162, 164, 170, 174, 200, 202</td>
</tr>
<tr>
<td>ILC</td>
<td>13, 212</td>
</tr>
<tr>
<td>NBK</td>
<td>45, 46, 75</td>
</tr>
<tr>
<td>RX2</td>
<td>11</td>
</tr>
<tr>
<td>RX4</td>
<td>10, 133</td>
</tr>
<tr>
<td>SYM</td>
<td>22, 53, 60, 76, 107</td>
</tr>
<tr>
<td>BFLD</td>
<td>135, 137</td>
</tr>
<tr>
<td>CAL1</td>
<td>16</td>
</tr>
<tr>
<td>EOF1</td>
<td>111</td>
</tr>
<tr>
<td>EOF2</td>
<td>156</td>
</tr>
<tr>
<td>EXPR</td>
<td>166, 176</td>
</tr>
<tr>
<td>FCNT</td>
<td>25, 117, 134</td>
</tr>
<tr>
<td>LBRK</td>
<td>75, 112, 114</td>
</tr>
<tr>
<td>LFLD</td>
<td>24, 124, 201, 224, 534</td>
</tr>
<tr>
<td>NSTG</td>
<td>17</td>
</tr>
<tr>
<td>SCN1</td>
<td>62</td>
</tr>
<tr>
<td>SCN2</td>
<td>155</td>
</tr>
<tr>
<td>STX4</td>
<td>123</td>
</tr>
<tr>
<td>TFLD</td>
<td>121, 141</td>
</tr>
<tr>
<td>XFLD</td>
<td>126, 63</td>
</tr>
<tr>
<td>BSCn1</td>
<td>171</td>
</tr>
<tr>
<td>BSCn2</td>
<td>27</td>
</tr>
<tr>
<td>CPAS1</td>
<td>136</td>
</tr>
<tr>
<td>CPAS2</td>
<td>23</td>
</tr>
<tr>
<td>ERASE</td>
<td>185</td>
</tr>
<tr>
<td>EXPW1</td>
<td>161, 165</td>
</tr>
<tr>
<td>EXPW2</td>
<td>172, 175</td>
</tr>
<tr>
<td>GENOP</td>
<td>206</td>
</tr>
<tr>
<td>LPAR</td>
<td>142, 157, 163, 173</td>
</tr>
<tr>
<td>LPAR2</td>
<td>150</td>
</tr>
<tr>
<td>MOSYM</td>
<td>101</td>
</tr>
<tr>
<td>PIVAR</td>
<td>204</td>
</tr>
<tr>
<td>RPAR2</td>
<td>153</td>
</tr>
<tr>
<td>RSCn1</td>
<td>35, 44, 116, 122</td>
</tr>
<tr>
<td>RSCn2</td>
<td>140, 145, 160</td>
</tr>
<tr>
<td>STFLD</td>
<td>56, 105, 113</td>
</tr>
<tr>
<td>TABBK</td>
<td>46, 75</td>
</tr>
<tr>
<td>TEOF1</td>
<td>20, 64, 110</td>
</tr>
<tr>
<td>COMCP</td>
<td></td>
</tr>
</tbody>
</table>

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP*: 00008 IN HUNDREDTHS OF MINUTES.
SUBROUTINE EXPR, EVALUATE EXPRESSION FOR COMPOP.

PCC  EXPR0010
COUNT 171  EXPR0020
LBL EXPR  BINARY CARD LABEL.  EXPR0040
ENTRY EXPR  ARITHMETIC EXPRESSION EVALUATOR.  EXPR0050

*  $EXPR IS CALLED BY,

*  TXS $EXPR,4
*  PZE LI,T,RI
*  PZE FLD

WHERE (FLD-LI) IS THE ADDRESS OF THE LEFT BREAK,
AND (FLD-RI) IS THE ADDRESS OF THE RIGHT BREAK.
EXPR TAKES A STRING OF SYMBOLS CONNECTED BY
- = OR / (S) ANG COMPILES THE RESULT IN
FLOATING POINT. IF T=0, THE RESULT IS PLACED IN
TEMPORARY STORAGE, OTHERWISE, RESULT IS IN AC. THE
SYMBOLIC FIELD IS MODIFIED ACCORDINGLY.

*  $EXPR OPERATES IN TWO PASSES. PASS1 USES $TERM
TO REDUCE THE EXPRESSION TO A SUMMATION. PASS2
PERFORMS THE SUMMATION. SOME OPTIMIZATION
IS DONE.

TRANSFER VECTOR
C0000  632551446060  TERM
C0001  473185219160  PIVAR
C0002  272545464760  GENOP
C0003  274562634660  GNSTC

LINKAGE DIRECTOR
C0004  000000000000
C0005  2567471516060

C0006  0634 00 4 00164  EXPR  SXA  RX4,4  SAVE IRS.  EXPR0250
C0007  0634 00 2 00165  SXA  RX2,2  **  EXPR0260
C0010  0634 00 1 00166  SXA  RX1,1  **  EXPR0270
C0011  -0500 00 4 00001  CAL  1,4  GET CONTROL WORD.  EXPR0280
C0012  0622 00 0 00054  STD  RSCN1+1  RIGHT BREAK INDEX, PASS 1.  EXPR0290
C0013  0622 00 0 00116  STD  RSCN2+1  RIGHT BREAK INDEX, PASS 2.  EXPR0300
C0014  0625 00 0 00171  STT  TAG  SAVE TAG FOR DECISION TO STORE.  EXPR0310
C0015  0621 00 0 00065  STA  TRMWO  SAVE LEFT BREAK INDEX FOR FIRST TERM.  EXPR0320
C0016  0734 00 1 00000  PAX  0,1  INDEX FOR LEFT BREAK.  EXPR0330
C0017  1 777777 1 00020  TXI  **+1,1,1,-1  INDEX OF FIRST WORD IN FLD.  EXPR0340
C0020  0634 00 1 00100  SXA  IWD1,1  SAVE FOR PASS2.  EXPR0350
C0021  -0500 00 4 00002  CAL  2,4  INSERT FLD ADDRESSES.  EXPR0360
C0022  0621 00 0 0035  STA  EPSA1  **  EXPR0370
C0023  0621 00 0 00162  STA  SCN2  **  EXPR0380
C0024  0621 00 0 00066  STA  BA1  **  EXPR0390
C0025  0621 00 0 00104  STA  BA2  **  EXPR0400
C0026  0402 00 0 00176  SUB  =1  **  EXPR0410
C0027  0621 00 0 00160  STA  BA3  **  EXPR0420
C0030  0600 00 0 00172  STZ  TEOF  RESET EOF MARK.  EXPR0430
C0031  0600 00 0 00170  STZ  LBRK  RESET LEFT BREAK.  EXPR0440
C0032  0600 00 0 00173  STZ  TSYM  RESET SYMBOL MARK.  EXPR0450
C0033  0600 00 0 00174  STZ  TSTSL  RESET */ MARK.  EXPR0460
SUBROUTINE EXPR, EVALUATE EXPRESSION FOR COMPOP.

00034 0020 00 0 00035  TRA  EPAS1  GO TO PASS 1 OF EXPR.  EXPRO470

SPACE 2
PASS1, REDUCE TO SUMMATION.

00035 -0500 00 0 0000  EPAS1  CAL  FLD,1  GET NEXT WORD IN FLD.
00036  0100 00 0 00053  TZE  RSCN1  IF ZERO, IGNORE.
00037 -0340 00 0 00202  LAS  +HO0000/  CHECK FOR /.
00038  0020 00 0 00042  TRA  **2  NO, SKIP.
00041  0020 00 0 00057  TRA  STSL1  YES, EXIT.
00042 -00340 00 0 00201  LAS  +HC0000/  CHECK FOR *.
00043  0020 00 0 00045  TRA  **2  NO, SKIP.
00044  0020 00 0 00057  TRA  STSL1  YES, EXIT.
00045 -0340 00 0 00200  LAS  +HO0000/  CHECK FOR -.
00046  0020 00 0 00050  TRA  **2  NO, SKIP.
00047  0020 00 0 00061  TRA  PLMI1  YES, EXIT.
00050 -0340 00 0 00177  LAS  +HO0000/  CHECK FCR +.
00051  0020 00 0 00053  TRA  **2  NO, SKIP.
00052  0020 00 0 00061  TRA  PLMI1  YES, EXIT.
00053 1 77777 1 00054  RSCN1 TXI **1,1,-1  COUNT WCROS.
00054 3 00000 1 00035  TXH  EPAS1,1,**  CHECK FCR EOF.
00055 -0625 00 0 00172  STL  TEOF  SET EOF MARK.
00056  0020 00 0 00061  TRA  PLMI1  AND TREAT AS +.
00057 -0625 00 0 00174  STSL1 STL TSTSL * OR /, SET MARK.
00060  0020 00 0 00053  TRA  RSCN1  RESUME SCAN.
00061 -0520 00 0 00174  PLMI1 NZT TSTSL END OF TERM, CHECK FOR */.
00062  0020 00 0 00070  TRA  NTSCL  NO, SKIP TERM.
00063 -06340 01 00065  SLD  TRMW0,1 SET RIGHT INDEX FOR TERM.
00064  0074 00 4 00000  TSX  &TERM,4  GO TO TERM.
00065  0 00000 0 00000  TRMWD PZE **,0,**  **.
00066  0 00000 0 00000  BAI PZE FLD **.
00067  0600 00 0 00174  STL  TSTSL RETURN FROM TERM, RESET STSL.
00070  06340 0 1 00065  NTSCL SXM TRMW0,1 SET NEXT LEFT INDEX.
00071 -0520 00 0 00172  NZT TEOF TEST FOR EOF.
00072  0020 00 0 00053  TRA  RSCN1  NO, RESUME SCAN.
00073  0020 00 0 00074  TRA  EPAS2  YES, GO TO PASS2.

SPACE 2
PASS2, COMPUTE SUM.

00074  0600 00 0 00173  EPAS2 STZ TSYM RESET SYMBOL MARK.
00075 -0500 00 0 00177  CAL  +H0000/  SET LBK**.
00076  0602 00 0 00170  SLW  LBK **.
00077 -0625 00 0 00175  STL FIRST SET FIRST ADDEND MARK.
00100  0774 00 1 00000  IWD1 AXI **,1 GET INDEX OF FIRST WORD IN FLD.
00101  0600 00 0 00172  STZ TECF RESET EC MARK.
00102 -0500 00 0 00000  SCN2 CAL FLD,1 GET NEXT WORD IN FLD.
00103  0100 00 0 00115  TZE RSCN2 IF ZERO, IGNORE.
00104  0600 00 0 00000  BA2 STZ FLD,1 ZERO FLD LOCATION.
00105 -0340 00 0 00200  LAS  +HO0000/  CHECK FCR -.
00106  0020 00 0 00110  TRA  **2 NO, SKIP.
00107  0020 00 0 00121  TRA  BRK YES, EXIT.

EXPRO480  EXPRO490  EXPRO500  EXPRO510  EXPRO520  EXPRO530  EXPRO540  EXPRO550  EXPRO560  EXPRO570  EXPRO580  EXPRO590  EXPRO600  EXPRO610  EXPRO620  EXPRO630  EXPRO640  EXPRO650  EXPRO660  EXPRO670  EXPRO680  EXPRO690  EXPRO700  EXPRO710  EXPRO720  EXPRO730  EXPRO740  EXPRO750  EXPRO760  EXPRO770  EXPRO780  EXPRO790  EXPRO800  EXPRO810  EXPRO820  EXPRO830  EXPRO840  EXPRO850  EXPRO860  EXPRO870  EXPRO880  EXPRO890  EXPRO900  EXPRO910  EXPRO920  EXPRO930  EXPRO940  EXPRO950  EXPRO960  EXPRO970  EXPRO980
SUBROUTINE EXPR, EVALUATE EXPRESSION FOR COMPOP.

00110 -0340 00 0 00177 LAS *H00000+ CHECK FOR +.
00111 0020 00 0 00113 TRA **2 NO, SKIP.
00112 0020 00 0 00121 TRA BRK YES, EXIT.
00113 0074 00 4 00001 TSX $PIVAR,4 SYMBOL, PLACE IN VARIABLE FIELD.
00114 -0625 00 0 00173 STL TSYM SET SYMBOL MARK.
00115 1 7777 1 00116 RSCN2 TXI *1,1,-1 COUNT WORDS.
00116 3 00000 1 00102 TXH RSCN2,1,** CHECK FOR EOF.
00117 -0625 00 0 00172 STL TECF SET EOF MARK.
00120 0020 00 0 00121 TRA BRK AND TREAT AS +-.

SPACE 2
ANALYZE BREAK.

00121 -0130 00 0 00000 BRK XCL GET LAST BREAK,
00122 0500 00 0 00170 CLA LBRK AND STORE THIS ONE IN LBRK.
00123 -0600 00 0 00170 STQ LBRK **
00124 0520 00 0 00175 ZET FIRST CHECK FOR FIRST ADDEND.
00125 0020 00 0 00137 TRA FBK YES, GO TO IT.
00126 0320 00 0 00177 ERA *H0000+ CHECK FOR +.
00127 0100 00 0 00131 TZE FAC YES, GO TO IT.
00130 0020 00 0 00134 TRA FSB NO, MUST BE -, GC TO IT.
00131 0074 00 4 00002 FAD TSX $GENOP,4 LBRK=+, OP=FAD.
00132 602621466060 BCI 1, FAD **
00133 0020 00 0 00153 TRA EDF2 CHECK FOR EOF.
00134 0074 00 4 00002 FSB TSX $GENOP,4 LBRK=-, OP=FSB.
00135 602622266060 BCI 1, FSB **
00136 0020 00 0 00153 TRA EDF2 CHECK FOR EOF.

SPACE 2
FIRST BREAK SECTION.

00137 -0520 00 0 00173 FBRK NZT TSYM CHECK FOR SYMBOL.
00140 0020 00 0 00153 TRA EDF2 NO SYMBOL, IGNORE UNARY OP.
00141 0600 00 0 00175 STZ FIRST FIRST ADDEND ENCOUNTERED.
00142 0322 00 0 00177 ERA *H0000+ CHECK FOR PLUS.
00143 0100 00 0 00145 TZE CLA YES, GO TO IT.
00144 0020 00 0 00150 TRA CLS NO, MUST BE -, GC TO IT.
00145 0074 00 4 00002 CLA TSX $GENOP,4 LBRK=++, OP=CLA.
00146 602343216600 BCI 1, CLA **
00147 0020 00 0 00153 TRA EDF2 CHECK FOR EOF.
00150 0074 00 4 00002 CLS TSX $GENOP,4 LBRK=--, OP=CLS.
00151 602343626060 BCI 1, CLS **
00152 0020 00 0 00153 TRA EDF2 CHECK FOR EOF.

SPACE 2
END-OF-FIELD AND RETURN SECTION.

00153 -0520 00 0 00172 EDF2 NZT TEOF TEST FOR EOF REACHED.
00154 0020 00 0 00115 TRA RSCN2 NO, RESUME SCAN.
SUBROUTINE EXPR, EVALUATE EXPRESSION FOR COMPO.

00155  0520 00 0 00171  ZET TAG CHECK TAG OF CONTROL WORD.  EXPR1490
00156  0020 00 0 00164  TRA RX4 NON-ZERO, LEAVE IN AC, RETURN.  EXPR1500
00157  0074 00 4 00003  TSX $GNSO,4 ZERO, GENERATE TEMPORARY STORAGE.  EXPR1510
00160  0602 00 1 77777  BA3 SLW FLO-1,1 INSERT IN FLO.  EXPR1520
00161  0074 00 4 00001  TSX $PIVAR,4 PLACE SYMBOL IN VARIABLE FIELD.  EXPR1530
00162  0074 00 4 00002  TSX $GENOP,4 FINAL OP IS STD.  EXPR1540
00163  60626346000  BCI 1, STD ..  EXPR1550
00164  0774 00 4 00000  RX4 AXT **,** RESTORE IRS.  EXPR1560
00165  0774 00 2 00000  RX2 AXT **,** ..  EXPR1570
00166  0774 00 1 00000  RX1 AXT **,** ..  EXPR1580
00167  0020 00 4 00003  TRA 3,4 RETURN TO CALLER.  EXPR1590

SPACE 2 STORAGE AND CONSTANTS.

00170  0 00000 0 0000  LBRK PZE LAST BREAK.  EXPR1600
00171  0 00000 0 0000  TAG PZE TAG OF CONTROL WORD.  EXPR1610
00172  0 00000 0 0000  TEDF PZE EOF MARK.  EXPR1620
00173  0 00000 0 0000  TSYM PZE SYMBOL MARK.  EXPR1630
00174  0 00000 0 0000  TSSTL PZE /* MARK.  EXPR1640
00175  0 00000 0 0000  FIRST PZE FIRST ADDEND MARK.  EXPR1650
00176  0 00000 0 0000  FLD EQU ** DUMMY SYMBOL FOR FLO.  EXPR1660
00177  0 00000 0 0000 0 0000 000000000001 LITERALS  EXPR1670
00178  0 00000 0 0000 0 0000 0000 LITERALS  EXPR1680
00179  0 00000 0 0000 0 0000 0000 LITERALS  EXPR1690
00180  0 00000 0 0000 0 0000 0000 LITERALS  EXPR1700
00181  0 00000 0 0000 0 0000 0000 LITERALS  EXPR1710
SUBROUTINE EXPR, EVALUATE EXPRESSION FOR COMPOP.  
POST PROCESSOR ASSEMBLY DATA

203 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

66  BA1  24
104  BA2  25
160  BA3  27
121  BRR  107,  112,  120
145  CLA  143
150  CLS  144
131  FAD  127
  0  FLD  35,  66,  102,  104,  160,  176
134  FSB  130
166  RX1  10
165  RX2  7
164  RX4  6,  156
171  TAG  14,  155
153  EOF2  133,  136,  140,  147,  152

6  EXPR
137  FBRK  125
100  IWL1  20
170  LBRK  31,  76,  122,  123
102  SCN2  23,  116
172  TECF  30,  55,  71,  101,  117,  153
  0  TERM  64
173  TSYM  32,  74,  114,  137
35  EPAS1  22,  34,  54
74  EPAS2  73
175  FIRST  77,  124,  141
  2  GENCP  131,  134,  145,  150,  162
  3  GNSIC  157
70  NSTSL  62
  1  PIVAR  113,  161
  61  PLM1  47,  52,  56
  53  RSCN1  12,  36,  60,  72
  115  RSCN2  13,  103,  154
  57  TSTLI  41,  44
  65  TRMWD  15,  63,  70
174  TSTSL  33,  57,  61,  67

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP..  000008 IN HUNDREDTHS OF MINUTES.
SUBROUTINE TERM, EVALUATE SIMPLE TERMS FOR EXPR.

PCG
COUNT 135
LBL TERM BINARY CARD LABEL.
ENTRY TERM EVALUATE TERM OF EXPRESSION.

$TERM IS CALLED BY,

* TSX $TERM,4
* PZE LI,G1
* PZE FLD
* WHERE (FLD-LI) IS THE ADDRESS OF THE LEFT BREAK,
* AND (FLD-RI) IS THE ADDRESS OF THE RIGHT BREAK.
* TERM TAKES A STRING OF SYMBOLS CONNECTED BY
* OR / (S) AND COMPILES THE RESULT IN FLOATING
* POINT. THE RESULT IS STORED IN TEMPORARY
* STORAGE, AND THE SPREAD FIELD IS MODIFIED ACCORDINGLY.
* SOME OPTIMIZATION IS DONE.

TRANSFER VECTOR
00000 473165215160
00001 272545464760
00002 255121622560
00003 274562634660

LINKAGE DIRECTR
00004 000000000000
00005 632591446060

00006 0634 00 4 00133 TERM SXA RX4,4 SAVE IRS.
00007 0634 00 2 00134 SXA RX2,2 **
00010 0634 00 1 00135 SXA RX1,1 **
00011 -0500 00 4 00001 CAL 1.4 GET CONTROL WORD.
00012 0622 00 0 00037 STD RSCAN+1 INDEX OF RIGHT BREAK.
00013 0734 00 1 00000 PAX 0.1 INDEX OF LEFT BREAK.
00014 1 7777 1 00015 TXI **1,1,-1 INDEX OF FIRST WORD IN FLD.
00015 -0500 00 4 00002 CAL 2.4 INSERT BUFFER ADDRESSES.
00016 0621 00 0 00024 STA SCAN **
00017 0621 00 0 00026 STA BA1 **
00020 0402 00 0 00140 SUB =1 **
00021 0621 00 0 00117 STA BA2 **
00022 0621 00 0 00127 STA BA3 **
00023 0600 00 0 00137 STZ LBK RESET LAST BREAK.
00024 -0500 00 1 00000 SCAN CAL FLO,1 GET NEXT WORD IN FLD.
00025 0100 00 0 00036 TZE RSCAN IF ZERO, IGNORE.
00026 0600 00 1 00000 BAI STZ FLC,1 ZERO FLD LOCATION.
00027 -0340 00 0 00142 LAS =HCO000/ CHECK FOR /.
00030 0026 00 0 0032 TRA **2 NO, SKIP.
00031 0020 00 0 0063 TRA SLASH YES, GO TO IT.
00032 -0340 00 0 00141 LAS =HCO000* CHECK FOR **
00033 0020 00 0 0035 TRA **2 NO, SKIP.
00034 0020 00 0 00041 TRA STAR YES, GO TO IT.
00035 0074 00 4 00000 TSX PIVAR,4 SYMBOL, PLACE IN VARIABLE FIELD.
00036 1 7777 1 00037 RSCAN TXI **1,1,-1 COUNT WORDS.
00037 3 00000 1 00024 TXH SCAN,1,** CHECK FOR EDF.
SUBROUTINE TERM, EVALUATE SIMPLE TERMS FOR EXPR.

SPACE 2
ENC-OF-FIELD AND RETURN SECTION.

C0105 0-0500 00 0 00137  EOF CAL LBRK EOF, GEI LBRK.
C0106 0100 00 0 00112  TZE EOF1 IF LBRK=0, GO TO IT.
C0107 0320 00 0 00141  ERA =H000000* CHECK, LBRK=**.
C0110 0100 00 0 00114  TZE EOF2 YES, GO TO IT.
C0111 0200 00 0 00124  TRA EOF3 NO, MUST BE /, GO TO IT.

C0112 0074 00 4 00002  EOF1 TSX $ERASE,4 LBRK=0, NO COMPILATION.
C0113 0020 00 0 00133  TRA RX4 GO TO RETURN.

C0114 0074 00 4 00001  EOF2 TSX $GENOP,4 LBRK=*, OP=FMP.
C0115 602644476606  BCI 1, FMP **
C0116 0074 00 4 00003  TSX $GENSTO,4 GENERATE TEMPORARY STORAGE.
C0117 0602 00 1 77777  BA2 SLW FLC-1,1 INSERT IN FLD.
C0120 0074 00 4 00000  TSX $PIVAR,4 PLACE SYMBOL IN VARIABLE FIELD.
C0121 0074 00 4 00001  TSX $GENOP,4 FINAL OP IS STO.
C0122 606263466060  BCI 1, STO **
C0123 0020 00 0 00133  TRA RX4 GO TO RETURN.

C0124 0074 00 4 00001  EOF3 TSX $GENOP,4 LBRK=1, OP=FDP.
C0125 602624476060  BCI 1, FDP **
C0126 0074 00 4 00003  TSX $GENSTO,4 GENERATE TEMPORARY STORAGE.
C0127 0602 00 1 77777  BA3 SLW FLC-1,1 INSERT IN FLD.
C0130 0074 00 4 00000  TSX $PIVAR,4 PLACE SYMBOL IN VARIABLE FIELD.
C0131 0074 00 4 00001  TSX $GENOP,4 FINAL OP IS STO.
C0132 606263506060  BCI 1, STO **

C0133 0774 00 4 00000  RX4 AXT **,4 RESTORE IRS.
C0134 0774 00 2 00000  RX2 AXT **,2 **
C0135 0774 00 1 00000  RX1 AXT **,1 **
C0136 0020 00 4 00003  TRA 3,4 RETURN TO CALLER.

SPACE 2
STORAGE AND CONSTANTS.

00137 0 00000 0 00000 LBRK PIE LAST BREAK CHARACTER.
00000 FLD EQU ** DUMMY SYMBOL FOR FLD.

END

LITERALS
C0140 000000000001
C0141 0000000000054
C0142 0000000000061

TERM0960
TERM0970
TERM0980
TERM0990
TERM1000
TERM1010
TERM1020
TERM1030
TERM1040
TERM1050
TERM1060
TERM1070
TERM1080
TERM1090
TERM1100
TERM1110
TERM1120
TERM1130
TERM1140
TERM1150
TERM1160
TERM1170
TERM1180
TERM1190
TERM1200
TERM1210
TERM1220
TERM1230
TERM1240
TERM1250
TERM1260
TERM1270
TERM1280
TERM1290
TERM1300
TERM1310
TERM1320
TERM1330
TERM1340
TERM1350
SUBROUTINE TERM, EVALUATE SIMPLE TERMS FOR EXPR.
POST PROCESSOR ASSEMBLY DATA

143 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>BA1</td>
</tr>
<tr>
<td>117</td>
<td>BA2</td>
</tr>
<tr>
<td>127</td>
<td>BA3</td>
</tr>
<tr>
<td>105</td>
<td>ECF</td>
</tr>
<tr>
<td>0</td>
<td>PLD</td>
</tr>
<tr>
<td>135</td>
<td>RX1</td>
</tr>
<tr>
<td>134</td>
<td>RX2</td>
</tr>
<tr>
<td>133</td>
<td>RX4</td>
</tr>
<tr>
<td>72</td>
<td>SL1</td>
</tr>
<tr>
<td>75</td>
<td>SL2</td>
</tr>
<tr>
<td>100</td>
<td>SL3</td>
</tr>
<tr>
<td>50</td>
<td>ST1</td>
</tr>
<tr>
<td>53</td>
<td>ST2</td>
</tr>
<tr>
<td>60</td>
<td>ST3</td>
</tr>
<tr>
<td>112</td>
<td>ECF1</td>
</tr>
<tr>
<td>114</td>
<td>ECF2</td>
</tr>
<tr>
<td>124</td>
<td>ECF3</td>
</tr>
<tr>
<td>137</td>
<td>LIBK</td>
</tr>
<tr>
<td>24</td>
<td>SCAN</td>
</tr>
<tr>
<td>41</td>
<td>STAR</td>
</tr>
<tr>
<td>6</td>
<td>TERM</td>
</tr>
<tr>
<td>2</td>
<td>ERASE</td>
</tr>
<tr>
<td>1</td>
<td>GENDP</td>
</tr>
<tr>
<td>3</td>
<td>GNSTO</td>
</tr>
<tr>
<td>0</td>
<td>PIVAR</td>
</tr>
<tr>
<td>36</td>
<td>RSCAN</td>
</tr>
<tr>
<td>63</td>
<td>SLASH</td>
</tr>
</tbody>
</table>

NO ERROR IN ABOVE ASSEMBLY.
TIME SPENT IN FAP.. 000006 IN HUNDREDTHS OF MINUTES.
CIOP... BUFFERED I/O PACKAGE FOR CAP. NO TAPE ERROR CHECKING.

PCC

COUNT 147

LBL CIOP BINARY CARD LABEL.

00003 ENTRY READ1 READ ONE RECORD ON INPUT TAPE.

00154 ENTRY PRINT WRITE ONE RECORD ON OUTPUT TAPE.

00300 ENTRY WCT1 WRITE ONE RECORD ON COLLATION TAPE.

00104 ENTRY REWIND END-FILE AND REWIND COLLATION TAPE.

00110 ENTRY READ2 READ ONE RECORD FROM COLLATION TAPE.

TRANSFER VECTOR

00000 256731556060 EXIT

LINKAGE DIRECTOR

00001 000000000000

00102 9121240160

00003 0634 00 2 00013 READ1 SXA RX2,2 SAVE IR2.

00004 -0500 00 4 00001 CAL 1,4 GET BUFFER ADDRESS.

00005 0621 00 0 00015 STA INCARD INSERT IN I/O COMMAND.

00006 0762 00 0 01202 RTDA 2 SELECT INPUT TAPE.

00007 0540 00 0 00015 RCHA INCAR START DATA CHANNEL.

00010 0060 00 0 00010 TCOA * WAIT FOR DATA TO ARRIVE.

00011 0774 00 2 00016 AXT WECF1,2 GET ERROR COMMENT LOCATION.

00012 0300 00 0 00214 TEFA ERROR CHECK FOR ERROR.

00013 0774 00 2 00000 RX2 AXT **,2 RESTORE IR2.

00014 0020 00 4 00002 TRA 2,4 RETURN TO CALLER.

00015 0 0016 0 00000 INCARD IOCD **,0,14 READ ONE 14 WORD RECORD AND STOP.

00016 0 0011 0 00017 WEOF1 IOCD **1,0,9 OUTPUT COMMENT IN CASE OF ERROR.

WCT1 SXA WX4,4 SAVE IRS.

00031 0634 00 4 00006 WCT1 SXA WX2,2 **.

00032 -0520 00 0 00023 NZT FSTART IS THIS THE FIRST CALL?

00033 0772 00 0 02023 REWB 3 YES, MAKE SURE TAPEREWIND.

00034 -0629 00 0 00223 STL FSTART SET MARKER.

00035 -0500 00 4 00001 CAL 1,4 GET CONTROL WORD.

00036 0625 00 0 00025 STT TAG GET TAG.

00037 -0520 00 0 00025 NZT TAG IF ZERO.

WCT1 WRITES A 14 WORD BUFFER ON TAPE B3. IF CONTROL WORD HAS A NON-ZERO TAG, COUNT IS OBTAINED FROM DECREMENT AND RECORD IS FILLED WITH BLANKS.

THIS ROUTINE IS BUFFERED, AND OUTPUT IS OVERLAPPED WITH COMPUTATION.

00030 0634 00 4 00067 WCT1 SXA WX4,4 SAVE IRS.

00031 0634 00 2 00066 SXA WX2,2 **.

00032 -0520 00 0 0023 NZT FSTART IS THIS THE FIRST CALL?

00033 0772 00 0 02203 REWB 3 YES, MAKE SURE TAPEREWIND.

00034 -0629 00 0 0223 STL FSTART SET MARKER.

00035 -0500 00 4 00001 CAL 1,4 GET CONTROL WORD.

00036 0625 00 0 00025 STT TAG GET TAG.

00037 -0520 00 0 00025 NZT TAG IF ZERO.
CIOP . . . BUFFERED I/O PACKAGE FOR CAP. NO TAPE ERRRR CHECKING.

0040 -0500 00 0 00263 CAL =14B17 ASSUME 14 WORDS IN BUFFER.
0041 -0734 00 0 00000 PDX 0,2 SAVE COUNT FOR MOVE OPERATION.
0042 0774 00 0 00022 AEX 1,8 MOVE COUNT TO ADDRESS.
0043 0361 00 4 00001 ACL 1,4 FCRM END ADDRESS.
0044 0621 00 0 00054 STA PKCUP INSERT IN PICKUP INSTRUCTION.
0045 0754 00 2 00000 PXA 0,2 SAVE IR2 IN AC.
0046 0774 00 0 00072 AXT WECCT,2 GET ADDRESS OF ERROR COMMENT.
0047 0061 00 0 00047 TCUB * WAIT FOR PREVIOUS WRITE TO FINISH.
0048 0000 00 0 02000 ETIB CHECK FOR END OF TAPE ON LAST WRITE.
0049 0020 00 0 00214 TRA ERROR END OF TAPE ENCOUNTRED, GO COMPLAIN.
004A 0734 00 2 00000 PAX 0,2 RESTORE IR2.
004B 0774 00 4 00016 AXT 14,4 SET MOVE COUNTER.
004C -0500 00 0 00000 PKCUP CAL **,2 MOVE DATA INTO OUTPUT BUFFER.
004D 0621 00 4 00262 SLW SLW WBUF**4,4 INSERT.
004E 2 00001 4 00060 TIX **,2,4,1 COUNT.
004F 0054 4 00054 TIX PKCUP**,2,1 INDEX, AND GET NEXT WORD.
0050 0060 4 00064 TXL ***,4,4,1 IS BUFFER FULL.
0051 0061 00 0 00264 CAL =H NO, FIL IT OUT WITH BLANKS.
0052 0062 00 6 00055 SLW SLW **
0053 2 00001 4 00061 TIX **,2,4,1 TEST FOR BUFFER FULL.
0054 0766 00 0 02200 WTBB 3 SELECT COLLATION TAPE.
0055 0540 00 0 00071 RCHB WECCT START CHANNEL.
0056 0774 00 2 00000 WX2 AXT **,2 RESTORE IR5.
0057 0774 00 4 00000 WX4 AXT ***,4 **
0058 0020 00 0 00002 TRA 2,4 RETURN TO CALLER.
0059 0000 00 0 02444 WECCT IODC WBUF**,0,14 WRITE A 14 WORD RECORD AND STOP.
005A 0000 00 0 00073 IODC WECCT,9,9 END OF TAPE REACHED WHILE WRITING COLLATION TAPE.
005B 0000 00 0 00011 BCI 9,END OF TAPE REACHED WHILE WRITING COLLATION TAPE.

0060 0770 00 0 02203 REWIND WEFR 3 WRITE AN END OF FILE.
0061 -0625 00 0 00224 STL TREW SET MARK FOR TAPE REMOVED.
0062 0772 00 0 02203 REWB 3 NOW REWINING TAPE.
0063 0020 00 0 00001 TRA 1,4 RETURN TO CALLER.
0064 0634 00 2 00122 READZ SZA RZ2,**,2 SAVE IR2.
0065 -0520 00 0 00224 NZT TREW HAS COLLATION TAPE BEEN REMOVED.
0066 0020 00 0 00125 TRA NREW NO, GC COMMENT.
0067 0500 00 4 00001 CAL 1,4 GET CONTROL WORD.
0068 0641 00 0 00212 STA IOTIN INSERT BUFFER ADDRESS IN I/O COMMAND.
0069 0621 00 0 00124 RTBB 3 READ SELECT COLLATION TAPE.
006A 0754 00 0 02223 RCMR 1,014 START CHANNEL.
006B 0774 00 2 00142 AXT WECCT,2 GET ADDRESS OF ERROR COMMENT.
006C 0001 00 0 01200 TCOB * WAIT FOR BUFFER TO FILL.
006D 0030 00 0 00124 TEFB ERROR HAS END OF FILE BEEN REACHED.
006E 0774 00 2 00000 RZ2 AXT **,2 ALL OK, RESTORE IR2.
006F 0020 00 0 00002 TRA 2,4 RETURN TO CALLER.
CIOP . . . BUFFERED I/O PACKAGE FOR CAP.  NO TAPE ERROR CHECKING.

<table>
<thead>
<tr>
<th>Line</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Address 3</th>
<th>Address 4</th>
<th>IOCT1 IORT</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00124</td>
<td>3 00016 0 00000</td>
<td>IOTin IORT</td>
<td>**,14</td>
<td>READ ONE RECORD AND STOP.</td>
<td>CIOP0900</td>
<td></td>
</tr>
<tr>
<td>00125</td>
<td>0074 00 4 00154</td>
<td>NREW TSX</td>
<td>PRINT,4</td>
<td>FORGOT TO REWIND, COMMENT.</td>
<td>CIOP0910</td>
<td></td>
</tr>
<tr>
<td>00126</td>
<td>0 00011 0 00131</td>
<td>PZE WREW,0,9</td>
<td>**</td>
<td>REWIND IT.</td>
<td>CIOP0920</td>
<td></td>
</tr>
<tr>
<td>00127</td>
<td>0074 00 4 00104</td>
<td>TSX RETRIN,4</td>
<td>READING IT.</td>
<td>CIOP0930</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00130</td>
<td>0020 00 0 00113</td>
<td>TRA READ,43</td>
<td>RETURN TO READ ROUTINE.</td>
<td>CIOP0940</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00131</td>
<td>005125212402</td>
<td>WNREW BCI</td>
<td>9,CREADZ CALLED BEFORE REWIND, COLLATION TAPE REWOUND.</td>
<td>CIOP0950</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00132</td>
<td>602321434325</td>
<td></td>
<td></td>
<td></td>
<td>CIOP0960</td>
<td></td>
</tr>
<tr>
<td>00133</td>
<td>246022252646</td>
<td></td>
<td></td>
<td></td>
<td>CIOP0970</td>
<td></td>
</tr>
<tr>
<td>00134</td>
<td>512560512566</td>
<td></td>
<td></td>
<td></td>
<td>CIOP0980</td>
<td></td>
</tr>
<tr>
<td>00135</td>
<td>31452473623</td>
<td></td>
<td></td>
<td></td>
<td>CIOP0990</td>
<td></td>
</tr>
<tr>
<td>00136</td>
<td>464343216331</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1000</td>
<td></td>
</tr>
<tr>
<td>00137</td>
<td>464560632147</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1010</td>
<td></td>
</tr>
<tr>
<td>00140</td>
<td>256051256646</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1020</td>
<td></td>
</tr>
<tr>
<td>00141</td>
<td>644524336060</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1030</td>
<td></td>
</tr>
<tr>
<td>00142</td>
<td>0 00011 0 00143</td>
<td>WEDFG IOCD</td>
<td>**,1,9</td>
<td>OUTPUT COMMENT IF ERROR.</td>
<td>CIOP1040</td>
<td></td>
</tr>
<tr>
<td>00143</td>
<td>002545246046</td>
<td>BCI</td>
<td>9,END OF FILE REACHED WHILE READING COLLATION TAPE.</td>
<td>CIOP1050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00144</td>
<td>266026314325</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1060</td>
<td></td>
</tr>
<tr>
<td>00145</td>
<td>605125212330</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1070</td>
<td></td>
</tr>
<tr>
<td>00146</td>
<td>252460663031</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1080</td>
<td></td>
</tr>
<tr>
<td>00147</td>
<td>432560512521</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1090</td>
<td></td>
</tr>
<tr>
<td>00150</td>
<td>243145276023</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1100</td>
<td></td>
</tr>
<tr>
<td>00151</td>
<td>464343216331</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1110</td>
<td></td>
</tr>
<tr>
<td>00152</td>
<td>464560632147</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1120</td>
<td></td>
</tr>
<tr>
<td>00153</td>
<td>253360606060</td>
<td></td>
<td></td>
<td></td>
<td>CIOP1130</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Address 3</th>
<th>Address 4</th>
<th>IOCT1 IORT</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00154</td>
<td>0634 00 2 00201</td>
<td>PRINT SXA</td>
<td>PX2,2</td>
<td>SAVE IR2.</td>
<td>CIOP1140</td>
<td></td>
</tr>
<tr>
<td>00155</td>
<td>0774 00 2 00204</td>
<td>AXT</td>
<td>WLEXN,2</td>
<td>GET ADDRESS OF ERROR COMMENT.</td>
<td>CIOP1150</td>
<td></td>
</tr>
<tr>
<td>00156</td>
<td>0500 00 0 00222</td>
<td>CLA</td>
<td>LNCNT</td>
<td>GET LNCOUNT.</td>
<td>CIOP1160</td>
<td></td>
</tr>
<tr>
<td>00157</td>
<td>0402 00 0 00262</td>
<td>SUB</td>
<td>1</td>
<td>LOWER BY ONE.</td>
<td>CIOP1170</td>
<td></td>
</tr>
<tr>
<td>00160</td>
<td>0610 00 0 00222</td>
<td>STO</td>
<td>LNCNT</td>
<td>RETURN.</td>
<td>CIOP1180</td>
<td></td>
</tr>
<tr>
<td>00161</td>
<td>-0120 00 0 00214</td>
<td>TMI</td>
<td>ERROR</td>
<td>EXIT IF TOO MANY LINES.</td>
<td>CIOP1190</td>
<td></td>
</tr>
<tr>
<td>00162</td>
<td>-0500 00 4 00001</td>
<td>CAL</td>
<td>1,4</td>
<td>GET CONTROL WORD.</td>
<td>CIOP1200</td>
<td></td>
</tr>
<tr>
<td>00163</td>
<td>0622 00 0 00203</td>
<td>STD</td>
<td>PID</td>
<td>INSERT COUNT IN I/O COMMAND.</td>
<td>CIOP1210</td>
<td></td>
</tr>
<tr>
<td>00164</td>
<td>-0734 00 2 00000</td>
<td>PDX</td>
<td>0,2</td>
<td>GET COUNT.</td>
<td>CIOP1220</td>
<td></td>
</tr>
<tr>
<td>00165</td>
<td>0771 00 0 00022</td>
<td>ARS</td>
<td>18</td>
<td>MOVE TO ADDRESS.</td>
<td>CIOP1230</td>
<td></td>
</tr>
<tr>
<td>00166</td>
<td>0361 00 4 00001</td>
<td>ACL</td>
<td>1,4</td>
<td>FORM END ADDRESS.</td>
<td>CIOP1240</td>
<td></td>
</tr>
<tr>
<td>00167</td>
<td>0621 00 0 00174</td>
<td>STA</td>
<td>GET</td>
<td>INSERT IN PICKUP.</td>
<td>CIOP1250</td>
<td></td>
</tr>
<tr>
<td>00170</td>
<td>0754 00 2 00000</td>
<td>PDX</td>
<td>0,2</td>
<td>GET COUNT.</td>
<td>CIOP1260</td>
<td></td>
</tr>
<tr>
<td>00171</td>
<td>0361 00 0 00203</td>
<td>ACL</td>
<td>PID</td>
<td>FORM BUFFER END ADDRESS.</td>
<td>CIOP1270</td>
<td></td>
</tr>
<tr>
<td>00172</td>
<td>0621 00 0 00175</td>
<td>STA</td>
<td>GIVE</td>
<td>INSERT IN STORE.</td>
<td>CIOP1280</td>
<td></td>
</tr>
<tr>
<td>00173</td>
<td>0060 00 0 00173</td>
<td>TCOA</td>
<td>*</td>
<td>WAIT FOR LAST PRINT TO FINISH.</td>
<td>CIOP1290</td>
<td></td>
</tr>
<tr>
<td>00174</td>
<td>-0500 00 2 00000</td>
<td>GET</td>
<td>CAL</td>
<td>**,2</td>
<td>MOVE DATA TO OUTPUT BUFFER.</td>
<td>CIOP0900</td>
</tr>
<tr>
<td>00175</td>
<td>0602 00 2 00000</td>
<td>GIVE</td>
<td>SLW</td>
<td>**,2</td>
<td>--</td>
<td>CIOP1300</td>
</tr>
<tr>
<td>00176</td>
<td>2 00001 2 00174</td>
<td>TIX</td>
<td>=-2,2,1</td>
<td>--</td>
<td>CIOP1310</td>
<td></td>
</tr>
<tr>
<td>00177</td>
<td>0766 00 0 01203</td>
<td>WTDA</td>
<td>3</td>
<td>WRITE SELECT OUTPUT TAPE.</td>
<td>CIOP1320</td>
<td></td>
</tr>
<tr>
<td>00200</td>
<td>0540 00 0 00203</td>
<td>RCHA</td>
<td>PID</td>
<td>START CHANNEL.</td>
<td>CIOP1330</td>
<td></td>
</tr>
<tr>
<td>00201</td>
<td>0774 00 2 00000</td>
<td>PX2</td>
<td>AXT</td>
<td>**,2</td>
<td>RESTORE IRS.</td>
<td>CIOP1340</td>
</tr>
<tr>
<td>00202</td>
<td>0020 00 4 00002</td>
<td>TRA</td>
<td>2,4</td>
<td>RETURN TO CALLER.</td>
<td>CIOP1350</td>
<td></td>
</tr>
<tr>
<td>00203</td>
<td>0 00000 0 00226</td>
<td>PID</td>
<td>IOCD</td>
<td>PBLFF,**</td>
<td>WRITE ONE RECORD AND STOP.</td>
<td>CIOP1360</td>
</tr>
<tr>
<td>00204</td>
<td>0 00007 0 00205</td>
<td>WLNEX IOCD</td>
<td>**,1,7</td>
<td>OUTPUT COMMENT IF ERROR.</td>
<td>CIOP1370</td>
<td></td>
</tr>
</tbody>
</table>
00205 004751462751  BCI  7,OPROGRAMMER OUTPUT EXCEEDS 300 RECORDS.  CIOP1300
00206 214444255160
00207 466463476643
00210 602567232525
00211 246260030000
00212 605125234691
00213 246233606060

00214 0634 00 2 00216  ERRCR SXA  **2,2  INSERT COMMAND LOCATION IN RCHA.  CIOP1310
00215 0766 00 0 01203  WTXA  3  WRITE SELECT OUTPUT TAPE.  CIOP1310
00216 0540 00 0 00000  RCHA  **  OUTPUT APPROPRIATE COMMENT.  CIOP1320
00217 0074 00 4 00000  CALL  EXIT  RETURN TO MONITOR FOR DUMP.  CIOP1330
00220 1 00000 0 00222
00221 0 1043 0 00001

00222 0 00000 0 00454  LNCNT PIE  300  MAXIMUM LINECOUNT.  CIOP1340
00223 0 00000 0 00000  FSTART PIE  FLAG FOR FIRST CALL TO WCT1.  CIOP1340
00224 0 00000 0 00000  FREW PIE  FLAG FOR COLLATION TAPE REWOUND.  CIOP1340
00225 0 00000 0 00000  TAG PIE  CIOP1340
00226 0 00000 0 00000  PBUFF BSS  14  PRINTER OUTPUT BUFFER.  CIOP1340
00227 0 00000 0 00000  WBUFF BSS  14  COLLATION TAPE OUTPUT BUFFER.  CIOP1340
00244  END

LITERALS
00262 000000000001
00263 000016000000
00264 606666666060

GENERAL ERROR ROUTINE.
MAKE SPECIFIED COMMENT, THEN RETURN TO MONITOR VIA EXIT.

STORAGE AND CONSTANTS.
CIOP . . . BUFFERED I/O PACKAGE FOR CAP. NO TAPE ERRCR CHECKING.
POST PROCESSOR ASSEMBLY DATA

265 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS

174 GET 167
203 PIC 163, 171, 200
201 PX2 154
13 RX2 3
55 SLW 62
225 TAG 36, 37
66 WX2 31
67 WX4 30
0 EXIT 217
175 GIVE 172
125 NREW 112
122 R2X2 110
224 TREW 105, 111
30 WCT1
214 ERRCR 12, 51, 121, 161
124 JOTIN 114, 116
71 IOWCT 65
222 LNCNT 156, 160
226 PRUFF 203
54 PCKUP 44, 57
154 PRINT 129
3 READ1
110 READ2 130
244 WBUFF 55, 71
72 WEOF 46
142 WEOF 117
16 WEOFJ 11
204 WLEN 155
191 WNREW 126
223 FSTART 32, 34
15 INCARD 5, 7
104 REWIND 127

NO ERROR IN ABCVE ASSEMBLY.
*TIME SPENT IN FAP.. 000007 IN HUNDREDTHS OF MINUTES.
TEST OF CAP, BEGIN ASSEMBLY.

CAP

REM THE FOLLOWING ARE ALL LEGAL CAP INSTRUCTIONS.

REM PROGRAM TO COUNT BITS IN AC.

50000 066000050016
50001 060000050021
50002 053400450020
50003 076000000001
50004 020000050013
50005 062000050022
50006 450000050021
50007 361000050017
50008 062000050021
50009 450000050022
50010 476500000001
50011 20001450003
50012 450000050021
50013 002100070000
50014 000000000000
50015 000001000000
50016 430000000044
50017 000000000000
50018 000001100000
50019 430000000044
50020 000000000000
50021 000000000000
50022 000000000000
50023 000000000000
50024 000000000000
50025 000000000000
50026 000000000000
50027 000000000000
50028 000000000000
50029 000000000000
50030 000000000000
50031 000000000000
50032 000000000000
50033 050000050005
50034 030000050003
50035 060100050013
50036 056000050003
50037 026000050005
50038 013100000000
50039 026000050013
50040 024100050015
50041 013100000000
50042 024100050016
50043 460000000000
50044 460000000000
50045 460000000000
50046 460000000000
50047 460000000000
50048 050000050011
50049 060100050000
50050 020000050036
50051 050000050022
50052 030200050015
50053 060100050000
50054 056000050020
50055 026000050013
50056 006000050010
50057 006000050010
50058 006000050010
50059 006000050010
50060 006000050010
50061 006000050010
50062 006000050010
50063 006000050010
50064 006000050010
50065 006000050010
50066 006000050010

COUNT

LDQ ZERO

STG COUNT

ZERO TEST CELLS

LDQ ZERO

COUNT 36 BITS.

CAL BITS

AND INCREMENT COUNT.

CAL WORD

RESTORE AC.

LGR 1

NEXT BIT.

TIX LOOP

INDEX.

DONE

OCTAL 002100070000

STOP WITH TRANSFER TO 7000 OCTAL.

REM STORAGE.

OCTAL 000000000000

INCREMENT OF ONE.

INT 1

ADDRESS IS 36.

ADJ X

TTEMP STORAGE FOR BIT COUNT.

THX S

TEMPORARY STORAGE FOR AC.

REM TEST OF CAP PSEUDO-OPS, AND FLAGS.

ILCD 8

ILLEGAL OPCODE.

TRA UNDEF

UNDERFIND SYMBOL.

INT 1,2,3,7,13A3,9

ERROR IN INTOP.

WMWX AEN

ILLEGAL OPCODE AND UNDEFINED SYMBOL.

COMP NO = YES + LOOP

CLA YES

FAD LOOP

ST0 NO

COMP COMP COUNT = LOOP * YES * NO / DONE / ZERO / ONE

LDQ LOOP

FMP YES

XCA

FMP NO

FDP DONE

FDP ZERO

FDP ONE

STQ TEM

CLA TEM

STO COUNT

USE OF SYMBOL DEFINED BY COMP.

COMP COUNT,1 = WORD,2 - DONE,4

CLA WORD,2

FSB DONE,4

STO COUNT,1

COMP WORD = (BIT0+THSX*(ONE+THSX*(ZERO+THSX*(DONE+THSX*NO)))))

TSTCAP00

TSTCAP01

TSTCAP02

TSTCAP03

TSTCAP04

TSTCAP05

TSTCAP06

TSTCAP07

TSTCAP08

TSTCAP09

TSTCAP10

TSTCAP11

TSTCAP12

TSTCAP13

TSTCAP14

TSTCAP15

TSTCAP16

TSTCAP17

TSTCAP18

TSTCAP19

TSTCAP20

TSTCAP21

TSTCAP22

TSTCAP23

TSTCAP24

TSTCAP25

TSTCAP26

TSTCAP27

TSTCAP28

TSTCAP29

TSTCAP30

TSTCAP31

TSTCAP32
RETURN FROM CAP, ENTRY POINT IS 50000.
Appendix B

PROGRAMS TO ALLOW USE OF CAP IN THE LABORATORY

This appendix contains FAP assembly listings of subprograms of the execution monitor and the I/Ø simulator used when CAP is used as a laboratory exercise. The listings are followed by a typical student output when running under the execution monitor. This output includes a storage map, CAP assembly listing, and postmortem.

Index to Appendix B

| Main program to call execution monitor | 104 |
| TESTS | 105 |
| TESTS) and associated entry points | 106 |
| RIP, WCT1, REWIND, READ2, PRINT, PPRØG, PCT1, READ1 | 143 |
| PRØG | 152 |
| Typical output listing | 155 |
MAIN PROGRAM FOR CAP.

TRANSFER VECTOR
00000 632562636260

TESTS
00001 0074 00 4 00000
00002
00002 -3 77777 7 77777
00003 -3 77777 7 77777
00004 -3 77777 7 77777

END

POST PROCESSOR ASSEMBLY DATA
5 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS
0 TESTS 1

NO ERROR IN ABOVE ASSEMBLY.
TIME SPENT IN FAP.. 000002 IN HUNDREDTHS OF MINUTES.
TESTS FOR CAP, SWITCH FOR INTERVAL TIMER.

PCC
COUNT 12

LBL TESTS
ENTRY TESTS

BINARY CARD LABEL.
INTERLUDE TO TESTS).

TRANSFER VECTOR
C0000 633163606060  TIT
C0001 632562836234  TESTS)

LINKAGE DIRECTOR
C0002 000000000000
C0003 632562836260

C0004 0760 00 0 00161  TESTS SWT 1  TEST SWITCH ONE FOR INTERVAL TIMER USE.
C0005 -0625 60 0 00000  STL* $TIT  SWITCH ONE UP, USE CORE CLOCK.
C0006 0772 00 0 01206  REWA 6  REWIND UPDATE INPUT TAPE.
C0007 0021 60 0 00001  TTR* $TESTS)  THEN GO DIRECTLY TO TESTS).

END

POST PROCESSOR ASSEMBLY DATA

10 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM

REFERENCES TO DEFINED SYMBOLS
0  TIT  5
4  TESTS
1  TESTS)  7

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP..  000002 IN HUNDREDTHS OF MINUTES.
<table>
<thead>
<tr>
<th>PCC</th>
<th>COUNT</th>
<th>LBL</th>
<th>TESTS</th>
<th>BINARY CARD LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00007</td>
<td>ENTRY</td>
<td>TESTS</td>
<td>PRIMAR NAME OF MONITOR</td>
<td></td>
</tr>
<tr>
<td>00702</td>
<td>ENTRY</td>
<td>BACK</td>
<td>RETURN POINT IN CASE OF ERROR</td>
<td></td>
</tr>
<tr>
<td>00235</td>
<td>ENTRY</td>
<td>TT</td>
<td>TEST LOCATION TO USE INTERVAL TIMER</td>
<td></td>
</tr>
<tr>
<td>00663</td>
<td>ENTRY</td>
<td>LST M</td>
<td>LEAVE-SELECT-TRAPPING MODE</td>
<td></td>
</tr>
<tr>
<td>01145</td>
<td>ENTRY</td>
<td>SX4</td>
<td>ENTRY TO SAVE IR4 FOR POST MORTEM</td>
<td></td>
</tr>
<tr>
<td>01146</td>
<td>ENTRY</td>
<td>SVCON</td>
<td>ENTRY TO SAVE CONSOLE FOR POST MORTEM</td>
<td></td>
</tr>
<tr>
<td>02407</td>
<td>ENTRY</td>
<td>W01</td>
<td>WRITE-OUTPUP-RAPE, A3</td>
<td></td>
</tr>
<tr>
<td>02520</td>
<td>ENTRY</td>
<td>NPRINT</td>
<td>ON LINE PRINT UNDER CARRIAGE CONTROL</td>
<td></td>
</tr>
<tr>
<td>01026</td>
<td>ENTRY</td>
<td>EPDR</td>
<td>ENTRY TO MARK ERROR POST MORTEM</td>
<td></td>
</tr>
<tr>
<td>01260</td>
<td>ENTRY</td>
<td>OCT</td>
<td>FULL WORD OCTAL-BCI CONVERTER</td>
<td></td>
</tr>
<tr>
<td>01173</td>
<td>ENTRY</td>
<td>OCTADR</td>
<td>OCTAL ADDRESS TO BCI CONVERTER</td>
<td></td>
</tr>
<tr>
<td>01233</td>
<td>ENTRY</td>
<td>COMADR</td>
<td>COMPLEMENT ADDRESS TO BCI CONVERTER</td>
<td></td>
</tr>
<tr>
<td>01207</td>
<td>ENTRY</td>
<td>TABBLK</td>
<td>TABLE FOR DELETION OF BLANKS WITH CR0</td>
<td></td>
</tr>
<tr>
<td>01301</td>
<td>ENTRY</td>
<td>BCDTAB</td>
<td>ENTRY TO REMOVE ILLEGAL BCI CHARACTERS</td>
<td></td>
</tr>
</tbody>
</table>
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION I, SETUP LOWER CORE AND PRINT STORAGE MAP.

00054 0361 60 0 00000 0 ACL $MOVIE) **
00055 0621 00 0 00073 STA SRCH COMPLETE ADDRESSES.
00056 0621 00 0 00076 STA SRCH+3 **
00057 0402 00 0 01327 SUB **
00060 0621 00 0 00136 STA MORG+4 **
00061 0400 00 0 01326 ADD =1 **
00062 0621 00 0 00104 STA MORG+2 **
00063 0400 00 0 01327 ADD =2 **
00064 0621 00 0 00102 STA MORG **
00065 0774 02 0 00002 AXT 2,2 SKIP FIRST TWO WORDS OF MOVIE.
00066 0774 01 0 0036 AXT NSYM,1 SET MAXIMUM NUMBER OF ENTRIES.
00067 0074 04 0 24207 TXS WOT,4 COMMENT.
00070 0 00006 0 00141 PZE XMAP01,0,6 **
00071 0074 00 0 02407 TXS WOT,4 COMMENT, HEADINGS.
00072 0 00004 0 00147 PZE XMAP02,0,4 **
00073 0500 02 0 00000 SRCR CLA **
00074 0402 00 0 0162 SUB **
00075 0100 00 0 01000 IZE MORG IF SO, EXIT TO PRINT NAME AND ORIGIN.
00076 0500 02 0 00000 CAL **
00077 0320 00 0 0164 ANA =077400077777 MASK OUT LARGEST POSSIBLE DECREMENT.
00078 0100 00 0 01000 IZE MORG IF ZERO, EXIT TO PROGRAM ORIGIN.
00079 1 00002 0 00073 TXI SRCR+2,2 NOT AN ORIGIN, BACK UP BY 2.
00080 0500 02 0 00000 MORG CAL **
00081 03067 STA ORIGIN+NSYM,1 STORE IN TABLE.
00082 0500 02 0 00000 CAL **
00083 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00084 0500 02 0 00000 CAL **
00085 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00086 0500 02 0 00000 CAL **
00087 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00088 0500 02 0 00000 CAL **
00089 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00090 0500 02 0 00000 CAL **
00091 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00092 0500 02 0 00000 CAL **
00093 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00094 0500 02 0 00000 CAL **
00095 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00096 0500 02 0 00000 CAL **
00097 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00098 0500 02 0 00000 CAL **
00099 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00100 0500 02 0 00000 MORG CAL **
00101 01325 STA MORG IF ZERO, EXIT TO PROGRAM ORIGIN.
00102 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00103 03067 STA ORIGIN+NSYM,1 STORE IN TABLE.
00104 0500 02 0 00000 CAL **
00105 0621 00 0 13015 IZE MORG IF SO, EXIT TO PRINT NAME AND ORIGIN.
00106 0500 02 0 00000 STA ENTRY+NSYM,1 STORE IN TABLE.
00107 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00108 0500 02 0 00000 CAL **
00109 0100 00 0 01000 STA ENTRY+NSYM,1 STORE IN TABLE.
00110 0500 02 0 00000 CAL **
00111 0601 00 0 03031 CAL =H(MAIN) MUST BE MAIN PROGRAM.
00112 0602 00 0 00154 SLW NAME+NSYM,1 STORE IN TABLE.
00113 0560 01 0 03067 LDQ ORIGIN+NSYM,1 GET ORIGIN AND CONVERT TO OCTAL-BCI.
00114 0074 00 4 01173 TXS OCTADR,4 **
00115 0600 00 0 00155 STQ XMAP03+2 INSERT IN COMMENT.
00116 0560 01 0 01325 LDQ ORIGIN+NSYM,1 GET ORIGIN AND CONVERT TO OCTAL-BCI.
00117 0074 00 4 01173 TXS OCTADR,4 **
00118 0600 00 0 00155 STQ XMAP03+3 INSERT IN COMMENT.
00119 0074 00 4 02407 TXS WOT,4 WRITE THIS COMMENT.
00120 0074 00 4 02407 TXS WOT,4 WRITE THIS COMMENT.
00121 0074 00 4 02407 TXS WOT,4 WRITE THIS COMMENT.
00122 0 00004 0 00153 PZE XMAP03,0,4 **
00123 0500 01 0 03031 CAL NAME+NSYM,1 IF THIS NAME IS TESTS, QUIT.
00124 0340 00 0 03161 LAS =TESTS **
00125 0200 00 0 00127 TRA ++
00126 0200 00 0 00133 TRA TRA NO, CONTINUE.
00127 -2 00001 1 00131 TRA TRA YES, EXIT.
00128 00004 2 00073 EXCM TXS SRCH+2,4 SKIP OVER FOUR AND TRY AGAIN.
00129 00004 2 00073 EXCM TXS SRCH+2,4 SKIP OVER FOUR AND TRY AGAIN.
00130 00004 2 00073 EXCM TXS SRCH+2,4 SKIP OVER FOUR AND TRY AGAIN.
00131 00004 2 00073 EXCM TXS SRCH+2,4 SKIP OVER FOUR AND TRY AGAIN.
00132 0 00004 0 0157 PZE XMAP04+1,4 **
00133 -0634 01 0 1447 TMAP SKD ILSUB,1 SAVE COUNT OF SYMBOL TABLE FOR POST MORTEM.
00134 0074 00 4 02407 TXS WOT,4 COMMENT, END OF SYMBOL TABLE.
00135 0 00004 0 0163 PZE XMAP05,0,4 **
00136 0061 00 0 00136 TCGT + WAIT FOR TIMING CHANNEL.
00137 0140 00 0 0175 TVD EVAL TURN OFF OVF LIGHT.
00138 0200 00 0 0175 TRA EVAL AND GO TO EVAL.
00139 0200 00 0 0175 TRA EVAL AND GO TO EVAL.
00141 006264224751 XMAP01 BCI 6,0 SUBPROGRAM STORAGE MAP FOLLOWS.
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 1, SETUP LOWER CORE AND PRINT STORAGE MAP.

00142 462751214460
00143 626346512127
00144 256044214760
00145 264643434666
00146 623360606060
00147 006060606060
00150 452144256060
00151 465131273145
00152 60254635170
00153 006060606060
00154 606060606060
00155 606060606060
00156 606060606060
00157 006270442246
00158 436063212243
00160 436063212243
00161 256025672325
00162 252425243360
00163 002545246046
00164 266062634651
00165 212725604421
00166 473360606060
00167 0021 00 0 00262 C(1) TTR TRAPR ETM TRAP RETURN.
00170 0021 00 0 00314 C(2) TTR STRR STR RETURN.
00171 0021 00 0 00524 C(7) TTR TIMR INTERVAL TIMER RETURN.
00172 0021 00 0 00040 C(8) TTR FPTR FPT RETURN.
00173 0021 00 0 00533 C(13) TTR SLR STOP-LOOP RETURN.
00174 0021 00 0 00627 C(5T) TTR IOTMR SELECT TRAP RETURN.

02223 T TAPENO B3B TAPE FOR TIMER.

TT1109060
TT110970
TT110980
TT110990
TT111000
TT111010
TT111020
TT111030
TT111040
TT111050
TT111060
TT111070
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 2, SET TRAPS, TIME AND EXIT TO CAP.

<table>
<thead>
<tr>
<th>TTL</th>
<th>SECTION 2, SET TRAPS, TIME AND EXIT TO CAP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00175 0074 00 4 02407</td>
<td>EVAL TSX WDT,4 EXECUTION COMMENT.</td>
</tr>
<tr>
<td>00176 0 00007 0 00241</td>
<td>PZE PCAP,0 7 10</td>
</tr>
<tr>
<td>00177 0074 00 4 02407</td>
<td>TSX WDT,4 BLANK LINE.</td>
</tr>
<tr>
<td>00200 0 00001 0 03160</td>
<td>PZE *H 0,1 10</td>
</tr>
<tr>
<td>00201 0074 00 4 00001</td>
<td>TSX $RIP,4 GO SET UP READ PROGRAMS.</td>
</tr>
<tr>
<td>00202 0520 00 0 00235</td>
<td>ZET TIT TEST FOR INTERVAL TIMER.</td>
</tr>
<tr>
<td>00203 0020 00 0 00207</td>
<td>TRA ITIM GO TO INTERVAL TIMER.</td>
</tr>
<tr>
<td>00204 0726 00 0 02223</td>
<td>WTBT TAPE TIMER, SELECT TIMER TAPE.</td>
</tr>
<tr>
<td>00205 0540 00 0 00237</td>
<td>RCHT TIME LOAD WITH TIME.</td>
</tr>
<tr>
<td>00206 0020 00 0 00213</td>
<td>TRA ESTM GO TO SET TRAP.</td>
</tr>
<tr>
<td>00207 0500 00 0 03136</td>
<td>ITIM CAL 300 SET INTERVAL TIMER FOR FIVE SECONDS.</td>
</tr>
<tr>
<td>00210 0760 00 0 00006</td>
<td>COM ** INSERT IN trap LOCATION.</td>
</tr>
<tr>
<td>00211 0601 00 0 00005</td>
<td>STD 5 CLEAR LOCATION RETURN.</td>
</tr>
<tr>
<td>00212 0600 00 0 00006</td>
<td>STZ 6 SET TRAP.</td>
</tr>
<tr>
<td>00213 0760 00 0 00005</td>
<td>ESTM SET TRAP.</td>
</tr>
<tr>
<td>00214 0500 00 0 00234</td>
<td>CAL ORG SET ORIGIN.</td>
</tr>
<tr>
<td>00215 0074 00 4 00002</td>
<td>TSX $CAP,4 GO TO CAP.</td>
</tr>
<tr>
<td>00216 0621 00 0 00236</td>
<td>SIA EXEC SAVE ENTRY POINT.</td>
</tr>
<tr>
<td>00217 0760 00 0 00007</td>
<td>LTM JUST IN CASE.</td>
</tr>
<tr>
<td>00220 0634 00 4 01166</td>
<td>SIA $RA,4 4 SAVE IR,4.</td>
</tr>
<tr>
<td>00221 0074 00 4 01146</td>
<td>TSX $SVCON,4 SAVE CONSOLE.</td>
</tr>
<tr>
<td>00222 0560 00 0 00236</td>
<td>LOQ EXEC CONVERT ENTRY TO OCTAL-BCI.</td>
</tr>
<tr>
<td>00223 0074 00 4 01173</td>
<td>TSX $DTCODR,4 **</td>
</tr>
<tr>
<td>00224 0754 00 0 00000</td>
<td>PZO 0,0 SHIFT AND OR TO COMMENT.</td>
</tr>
<tr>
<td>00225 0763 00 0 00036</td>
<td>LGL 30 10</td>
</tr>
<tr>
<td>00226 0602 00 0 00257</td>
<td>ORS WEP+7 **</td>
</tr>
<tr>
<td>00227 0130 00 0 00000</td>
<td>XCL **</td>
</tr>
<tr>
<td>00230 0602 00 0 00260</td>
<td>ORS WEP+8 **</td>
</tr>
<tr>
<td>00231 0074 00 4 02407</td>
<td>TSX WDT,4 COMMENT, ENTRY POINT.</td>
</tr>
<tr>
<td>00232 0074 00 4 00250</td>
<td>PZE WEP+0,9 10</td>
</tr>
<tr>
<td>00233 0020 00 0 00702</td>
<td>TRA BACK NO PROVISION FOR EXECUTION FOR NOW.</td>
</tr>
<tr>
<td>00234 000000005000</td>
<td>ORG OCT 50000 CAP PROGRAM ORIGIN.</td>
</tr>
<tr>
<td>00235 0 00000 0 00000</td>
<td>TIT PZE LOCATION TO TEST FOR INTERVAL TIMER.</td>
</tr>
<tr>
<td>00236 0 00000 0 00000</td>
<td>EXEC PZE DUMMY TRA TO ASSEMBLED PROGRAM.</td>
</tr>
<tr>
<td>00237 0 06050 0 17130</td>
<td>TIME DOMP -25000,0,25000 TIMED FOR ABOUT 5 SECONDS.</td>
</tr>
<tr>
<td>00238 -1 06050 0 17130</td>
<td>IOCT -25000,0,25000 **</td>
</tr>
<tr>
<td>00239 01600606060</td>
<td>TCAP BCI 7,1 TEST OF CAP, BEGIN ASSEMBLY.</td>
</tr>
<tr>
<td>00240 6060606060</td>
<td>WEP BCI 9,0 RETURN FROM CAP, ENTRY POINT IS 00000.</td>
</tr>
<tr>
<td>00241 6060606060</td>
<td>WEZ ASSEMBLE.</td>
</tr>
<tr>
<td>00242 6060606060</td>
<td>224 373036060</td>
</tr>
<tr>
<td>00243 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00244 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00245 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00246 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00247 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00248 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00249 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00250 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00251 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00252 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00253 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00254 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00255 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00256 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00257 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00258 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00259 6060606060</td>
<td>222 4323214773</td>
</tr>
<tr>
<td>00260 6060606060</td>
<td>222 4323214773</td>
</tr>
</tbody>
</table>

TTL20000
TTL20010
TTL20020
TTL20030
TTL20040
TTL20050
TTL20060
TTL20070
TTL20080
TTL20090
TTL20100
TTL20110
TTL20120
TTL20130
TTL20140
TTL20150
TTL20160
TTL20170
TTL20180
TTL20190
TTL20200
TTL20210
TTL20220
TTL20230
TTL20240
TTL20250
TTL20260
TTL20270
TTL20280
TTL20290
TTL20300
TTL20310
TTL20320
TTL20330
TTL20340
TTL20350
TTL20360
TTL20370
TTL20380
TTL20390
TTL20400
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 3, TRAP RETURNS AND LSTM.

QUIT IODC 0,0,0 COMMAND TO DISCONNECT CHANNEL.

SPACE 2
TRAPPING MODE RETURN.

00262 -0760 00 0 00007 TRAPR LSTM RETURN FROM TRANSFER TRAP.
00263 0634 00 4 01166 SXA IR4,4 SAVE IR4.
00264 0074 00 4 01146 TSX SVCN,4 SAVE CONSOLE.
00265 0520 00 0 00000 NZT 0 IF LOCATION O NOT SET.
00266 0020 00 0 00361 TRA SEGR MUST BE A WILD TRANSFER.
00267 0560 00 0 00000 LDQ 0 GET ADDRESS OF TRANSFER INSTRUCTION.
00270 0074 00 4 01173 TSX OCTADR,4 CONVERT TO CCTAL-BCD.
00271 0754 00 0 00000 PXD 0,0 ...
00272 0763 00 0 00022 LGL 18 SHIFT AND OR TO COMMENT.
00273 0602 00 0 00307 ORS XTRAP+5 ...
00274 0130 00 0 00000 XGL ...
00275 0602 00 0 00310 ORS XTRAP+6 ...
00276 0074 00 4 02407 TSX WOT,4 COMMENT.
00277 0 00012 1 00302 PZE XTRAP,1,10 ...
00300 -0625 00 0 01026 STL EPB,1 SET TO GIVE ERROR POST MORTEM.
00301 0200 00 0 00000 TRA BACK EXIT TO POST MORTEM SECTION.

00302 006351214562 XTRAP BCI 9,0TRANSFER INSTRUCTION IN LOCATION 00000 HAS BEEN TRAPP
00303 262551603145
00304 626351642363
00305 314645603145
00306 604346232163
00307 314645600000
00310 000000603021
00311 626022252545
00312 606351214747
00313 252433606060 BCI 1,ED.

SPACE 2
STR RETURN.

00314 -0760 00 0 00007 STRR LSTM RETURN FOR TRAPPED STR.
00315 0634 00 4 01166 SXA IR4,4 SAVE IR4.
00316 0074 00 4 01146 TSX SVCN,4 SAVE CONSOLE.
00317 0520 00 0 00000 NZT 0 IF LOCATION O NOT SET.
00320 0020 00 0 00361 TRA SEGR MUST BE A WILD TRANSFER.
00321 0534 00 4 00000 LXA 0,4 GET A(STR)+1.
00322 1 77777 4 00323 TXI *1,4,-1 DECREMENT IT.
00323 0634 00 4 00324 SXA *1,4 ...
00324 0560 00 0 00000 LDQ 0 CHECK FCR PROGRAMMED STR.
00325 -0754 00 0 00000 PXD 0,C ...
00326 -0763 00 0 00003 LGL 3 ...
00327 0402 00 0 03131 SUB #5 ...
00330 0100 00 0 00334 TZE #4 YES.
00331 -0500 00 0 00000 CAL 0 IF NOT, STOP OR LOOP.
00332 0621 00 0 00610 STA SADR
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM
SECTION 3, TRAP RETURNS AND LSTM

00333 0020 00 0 00541  TRA  SLR1  ...  TT130430
00334 0754 00 0 00000  PXA  0,4  PROGRAMMED STR.
00335 -0130 00 0 00000  XCL  ...  TT130440
00336 0074 00 0 01173  TSX  OCTADR,4  CONVERT TO OCTAL-BCD.
00337 -0754 00 0 00000  PXD  0,0  ...  TT130460
00340 -0763 00 0 00014  LGL  12  SHIFT AND OR TO COMMENT.
00341 -0602 00 0 00354  ORS  XSTR+4  ...  TT130480
00342 -0130 00 0 00000  XCL  ...  TT130490
00343 -0602 00 0 00355  ORS  XSTR+5  ...  TT130500
00344 0074 00 0 02407  TSX  WDT,4  COMMENT.
00345 0 00011 1 00350  PZE  XSTR+1,9  ...  TT130510
00346 -0625 00 0 01026  STL  EPMR  SET TO GIVE ERROR POST MORTEM.
00347 0020 00 0 00702  TRA  BACK  EXIT TO POST MORTEM SECTION.  TT130520
00350 006263510631  XSTR BCI  9,0STR INSTRUCTION IN LOCATION 00000 HAS BEEN TRAPPED.  TT130530
00351 456263516423
00352 633146456031
00353 456043462321
00354 633146456000
00355 000000000030
00356 262602222526
00357 456063512147
00360 472524336060

SPACE 2
ENTRY FOR SEQUENCING ERROR.

00361 0074 00 0 02407  SEQR  TSX  WDT,4  COMMENT FOR BAD CALLING SEQUENCE.
00362 0 00017 1 00365  PZE  XSEQR,1,15  ...  TT130590
00363 -0625 00 0 01026  STL  EPMR  SET TO GIVE ERROR POST MORTEM.
00364 0020 00 0 00702  TRA  BACK  EXIT TO POST MORTEM SECTION.  TT130600
00365 006351214562
00366 262551606346
00367 60436662551
00370 60236512533
00371 604751462221
00372 244370603151
00373 046045646360
00374 512562256373
00375 604651606321
00376 276044316262
00377 314527602651
00400 464460635121
00401 456226255160
00402 314562635164
00403 236331464533

SPACE 2
FLOATING POINT TRAP RETURN.

00404 -0520 00 0 00000  FPTR NZT  0  IF LOCATION 0 NOT SET,
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 3, TRAP RETURNS AND LSTM.

00405  0021  00  00425  TTR  TR8  MUST BE A WILD TRANSFER.  T130730
00406  0640  00  00473  STI  SSI  SAVE THE INDICATORS.  T130740
00407  0641  00  00000  LOI  0  GET SPILL CODE.  T130750
00410  -0054  00  00004  LFT  4  CHECK FOR OVERFLOW.  T130760
00411  0021  00  00431  TTR  FPER  OVERFLOW, EXIT.  T130770
00412  -0054  00  00002  LFT  2  AC UNDERFLOW.  T130780
00413  0500  00  03125  CLA  =0  YES, RESET AC.  T130790
00414  -0054  00  00001  LFT  1  MQ UNDERFLOW.  T130800
00415  0560  00  03125  LOU  =0  YES, RESET MQ.  T130810
00416  0641  00  00473  LOI  SSI  RESTORE ORIGINAL INDICATORS.  T130820
00417  0634  00  00422  SKA  **+3,4  SAVE IR4.  T130830
00420  0534  00  00000  LXA  0,4  PICKUP 0.  T130840
00421  0634  00  00424  SKA  **+3,4  INSERT RETURN ADDRESS.  T130850
00422  0774  00  00000  AXT  **+3,4  RESTORE IR4.  T130860
00423  0600  00  00000  STZ  0  CLEAR LOCATION 0.  T130870
00424  0021  00  00000  TTR  **  RETURN TO CALLER.  T130880
00425  -0760  00  00007  TR8 LTM  JUST IN CASE.  T130900
00426  0634  00  01166  SKA  IR4,4  SAVE IR4 FOR POST MORTEM.  T130910
00427  0074  00  01146  TSA  SVCON,4  GO SAVE CONSOLE AND RESET TRAPS.  T130920
00430  0020  00  00361  TRA  SEQ.  GO TO SEQR ROUTINE.  T130930
00431  -0760  00  00007  FPER LTM  JUST IN CASE.  T130940
00432  0634  00  01166  SKA  IR4,4  SAVE IR4.  T130950
00433  0441  00  00473  LOI  SSI  RESTORE INDICATORS.  T130960
00434  0074  00  01146  TSA  SVCON,4  SAVE CONSOLE.  T130970
00435  0641  00  00000  LOI  0  GET SPILL CODE AND ADDRESS OF INSTRUCTION.  T130980
00436  -0056  00  00010  LNT  10  DIVISION ERROR.  T131000
00437  0020  00  00443  TRA  **4  NO, SKIP.  T131010
00440  0074  00  02407  TSA  WOT,4  YES, COMMENT.  T131020
00441  00000  1 00474  PZE  WOVR,1,3  **  T131030
00442  0020  00  00453  TRA  SPER  EXIT.  T131040
00443  -0056  00  00002  LNT  2  ACCUMULATOR OVERFLOW.  T131050
00444  0020  00  00447  TRA  **3  NO, SKIP.  T131060
00445  0074  00  02407  TSA  WOT,4  YES, COMMENT.  T131070
00446  00000  1 00477  PZE  WACO,1,4  **  T131080
00447  -0056  00  00001  LNT  1  MQ OVERFLOW.  T131090
00450  0020  00  00453  TRA  SPER  NO, EXIT.  T131100
00451  0074  00  02407  TSA  WOT,4  YES, COMMENT.  T131110
00452  00000  1 00503  PZE  WMQR,1,5  **  T131120
00453  -0500  00  00000  SPER  CAL  0  GET SPILL CODE.  T131130
00454  -0320  00  03141  ANA  =000017000000  MASK OUT JUNK.  T131140
00455  -0765  00  00025  LGR  21  CONVERT TO OCTAL.  T131150
00456  0767  00  00003  ALS  3  SHIFT AND OR TO COMMENT.  T131160
00457  -0763  00  00003  LGL  3  **  T131170
00460  -0602  00  00522  ORS  XFPT+10  **  T131180
00461  -0500  00  00000  CAL  0  GET ADDRESS OF TRAPPED INSTRUCTION.  T131190
00462  0402  00  03126  SUB  =1  **  T131200
00463  -0130  00  00000  XCL  **  T131210
00464  0074  00  01173  TSX  OCTADR,4  CONVERT TO OCTAL-BCD.  T131220
00465  -0130  00  00000  XCL  OR TO COMMENT.  T131230
00466  -0602  00  00517  ORS  XFPT+7  **  T131240
00467  0074  00  02407  TSX  WOT,4  COMMENT.  T131250
00470  00014  1 00510  PZE  XFPT+1,12  **  T131260
00471  -0625  00  01026  STL  EPMR  SET TO GIVE ERROR POST MORTEM.  T131270
TESTS FOR CAP., MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 3, TRAP RETURNS AND LST.

00472 0020 00 00702  TRA  BACK  EXIT TO POST MORTEM SECTION.
00473 000000 00000  SSI  PZE  TEMPORARY STORAGE FOR SI.
00474 002431653162  WDVER BCI 3,0DIVISION ERROR.
00475 314645602551  WACO BCI 4,0ACUMULATOR OVERFLOW.
00476 514651336360  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00477 002123236444  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00500 644321633451  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00501 604665255126  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00503 434666336360  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00504 474331255140  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00505 506446663125  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00506 456360466525  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00507 51264366613  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00508 002643364263  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00510 314527604746  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00511 314563606247  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00512 314363606247  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00513 236451125224  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00515 603145604346  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00516 232163314645  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00517 600000000000  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00518 736062473143  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00519 436023462425  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00520 603162600000  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.
00523 336060606060  WMQO BCI 5,0MULTIPLIER-QUOTIENT OVERFLOW.

SPACE 2
INTERVAL TIMER RETURN.

00524 -0760 00 000007  TIMR LTM  JUST IN CASE.
00525 0634 00 001166  SXA  IR4,4  SAVE IR4 FOR POST MORTEM.
00526 0074 00 001166  TSX  S4CON,4  SAVE CONSOLE AND RESET TRAPS.
00527 -0500 00 000006  CAL  6  PICKUP TRAP LOCATION.
00530 0100 00 00361  TZE  SEQR  IF ZERO, MUST BE WILD TRANSFER.
00531 0621 00 00610  STA  SADR  GO TO ANALYZE.
00532 0020 00 00541  TRA  SLR  STOP-LOOP TRAPPED RETURN.

SPACE 2
TAPE TIMER RETURN.

00533 -0760 00 000007  SLR LTM  STOP-LOOP TRAPPED RETURN.
00534 0634 00 001166  SXA  IR4,4  SAVE IR4.
00535 0074 00 001166  TSX  SCON,4  SAVE CONSOLE.
00536 -0500 00 000014  CAL  12  SAVE TRAP LOCATION.
00537 0100 00 00361  TZE  SEQR  IF ZERO, MUST BE WILD TRANSFER.
00540 0621 00 00610  STA  SADR  GO TO ANALYZE.
00541 -0500 00 00610  SLR  CAL 1  MASK IT.
00542 -0320 00 00312  TZE  PF  IF HIR OR FOR, CALL IT A STOP.
00543 0100 00 00571  TZE  PSTOP  SAVE TRAP LOCATION.
00544 -0500 00 00610  CAL  SADR  REDUCE C(ILC) BY 1.
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 3, TRAP RETURNS AND LSM.

00546 0621 00 0 00610 STA SACR **
00547 -0500 60 0 00610 CAL SADR GET TRAPPED INSTRUCTION.
00550 -0320 00 0 03165 ANA =077700000000 MASK IT.
00551 0774 00 4 00004 AKT 4,4 COMPARE WITH TRUE STOPS.
00552 -0340 00 0 00610 LAS PSTOP+4,4 **
00553 0020 00 0 00555 TRA **2 NO.
00554 0220 00 0 00571 TRA PSTOP TRUE STOP.
00555 200001 4 00952 TIX =-3,4,1 NO, INDEX.

00556 0560 00 0 00610 LOOP LDQ SADR
00557 0074 00 4 01173 TSX OCTADR+4 CONVERT TO OCTAL BCD.
00558 -0754 00 0 00000 PXD 0,0 **
00559 -0763 00 0 00030 LCL 24 SHIFIT AND OR TO COMMENT.
00560 -0602 00 0 00617 ORS XLCOP+6 **
00561 -0130 00 0 00000 XCL **
00562 -0602 00 0 00620 ORS XLCOP+7 **
00563 0074 00 4 02407 TSX WOT,4 COMMENT.
00564 00010 0 00611 PZE XLCOP+1,8 **
00565 -0625 00 0 01026 STI EPMR SET TO GIVE ERROR POST MORTEM.
00567 0020 00 0 00702 TRA BACK EXIT TO POST MORTEM SECTION.

00571 0560 00 0 00610 PSTOP LDQ SACR TRUE STOP, GET ADDRESS OF TRAPPED
00572 0074 00 4 01173 TSX OCTADR+4 INSTRUCTION AND CONVERT TO OCTAL-BCD.
00573 -0754 00 0 00000 PXD 0,0 **
00574 -0763 00 0 00036 LCL 30 SHIFT AND OR TO COMMENT.
00575 -0602 00 0 00625 ORS XPSTOP+4 **
00576 -0130 00 0 00000 XCL **
00577 -0602 00 0 00626 ORS XPSTOP+5 **
00560 0074 00 4 02407 TSX WOT,4 COMMENT.
00561 00000 1 00621 PZE XPSTOP+1,6 **
00562 -0625 00 0 01026 STI EPMR SET TO GIVE ERROR POST MORTEM.
00563 0020 00 0 00702 TRA BACK EXIT TO POST MORTEM SECTION.

00604 0220 00 0 00000 PSTOPS OVH - LIST OF STOPS.
00605 0224 00 0 00000 VDH -";" **
00606 0240 00 0 00000 FDH - **
00607 0420 00 0 00000 MPR -
00610 0 00000 0 00000 SADR PZE ADDRESS OF STOP.
00611 004751462221 XLOOP BCI 8,0PROGRAM ENDLESS LCOOP AROUND LOCATION 00000.
00612 224325602546 XPSTOP BCI 6,0PROGRAM STOP AT LOCATION 00000.
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00627 - 0760 00 0 00007</td>
<td>IOTMR LTM</td>
<td>GET (ADDRESS+1) OF TRAPPED INSTRUCTION.</td>
</tr>
<tr>
<td>00630 0064 00 0 01166</td>
<td>SXA</td>
<td>SAVE IR4.</td>
</tr>
<tr>
<td>00631 0074 00 0 01146</td>
<td>TX</td>
<td>SAVE THE CONSOLE.</td>
</tr>
<tr>
<td>00632 - 0760 00 0 00010</td>
<td>LSNM</td>
<td>JUST IN CASE.</td>
</tr>
<tr>
<td>00633 0534 00 0 40000</td>
<td>TXA</td>
<td>SET TO ADDRESS OF TRAPPED INSTRUCTION.</td>
</tr>
<tr>
<td>00634 17777 4 00635</td>
<td>TXI</td>
<td>JUST IN CASE.</td>
</tr>
<tr>
<td>00635 0754 00 0 00000</td>
<td>PXA</td>
<td>TO AC.</td>
</tr>
<tr>
<td>00636 - 0130 00 0 00000</td>
<td>TXS</td>
<td>TO MG.</td>
</tr>
<tr>
<td>00637 0074 00 0 01173</td>
<td>ZAC</td>
<td>INSERT IN COMMENT.</td>
</tr>
<tr>
<td>00640 - 0754 00 0 00000</td>
<td>XCL</td>
<td>CONVERT TO OCTAL-BCD.</td>
</tr>
<tr>
<td>00641 - 0763 00 0 00022</td>
<td>ORS</td>
<td>SET TO ADDRESS OF TRAPPED INSTRUCTION.</td>
</tr>
<tr>
<td>00642 - 0602 00 0 00661</td>
<td>ORS</td>
<td>INSERT IN COMMENT.</td>
</tr>
<tr>
<td>00643 - 0130 00 0 00000</td>
<td>ORS</td>
<td>CONVERT TO OCTAL-BCD.</td>
</tr>
<tr>
<td>00644 - 0602 00 0 00662</td>
<td>ORS</td>
<td>COMMENT.</td>
</tr>
<tr>
<td>00645 0074 00 0 02407</td>
<td>TXS</td>
<td>SET TO GIVE ERROR POST MORTEM.</td>
</tr>
<tr>
<td>00646 0 00012 1 00631</td>
<td>PZE</td>
<td>SET TO GIVE ERROR POST MORTEM SECTION.</td>
</tr>
<tr>
<td>00647 - 0625 00 0 01026</td>
<td>STL</td>
<td>EXIT TO POST MORTEM SECTION.</td>
</tr>
<tr>
<td>00650 0020 00 0 00702</td>
<td>TRA</td>
<td>EXIT TO POST MORTEM SECTION.</td>
</tr>
<tr>
<td>00651 002163632544</td>
<td>XI0BD BCI</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00652 476360634660</td>
<td>XIOBD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00653 666225603143</td>
<td>XI0BD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00654 432527214360</td>
<td>XI0BD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00655 316146003145</td>
<td>XI0BD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00656 626351642363</td>
<td>XI0BD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00657 314645603145</td>
<td>XI0BD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00660 60434523163</td>
<td>XI0BD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00661 31464560000</td>
<td>XI0BD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
<tr>
<td>00662 000000336060</td>
<td>XI0BD</td>
<td>9,0ATTEMPT TO USE ILLEGAL I/O INSTRUCTION IN LOCATION OO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00663 - 0600 00 0 00677</td>
<td>LSTM STQ</td>
<td>SAVE MQ.</td>
</tr>
<tr>
<td>00664 - 0760 00 0 00010</td>
<td>LSTM</td>
<td>JUST IN CASE.</td>
</tr>
<tr>
<td>00665 0560 00 0 40001</td>
<td>LDQ</td>
<td>SAVE TRAP RETURN.</td>
</tr>
<tr>
<td>00666 - 0600 00 0 00700</td>
<td>STQ</td>
<td>SET TRAP.</td>
</tr>
<tr>
<td>00667 0560 00 0 00700</td>
<td>LDQ</td>
<td>SET TRAP.</td>
</tr>
<tr>
<td>00670 - 0600 00 0 40001</td>
<td>STQ</td>
<td>SAVE TRAP RETURN.</td>
</tr>
<tr>
<td>00671 0766 00 0 01204</td>
<td>MT0A</td>
<td>NIL COMMAND.</td>
</tr>
<tr>
<td>00672 0540 00 0 00261</td>
<td>RCH</td>
<td>RESTORE ORIGINAL TRAP.</td>
</tr>
<tr>
<td>00673 0560 00 0 00700</td>
<td>RLSTM LDQ</td>
<td>RESTORE ORIGINAL MQ.</td>
</tr>
<tr>
<td>00674 - 0600 00 0 40001</td>
<td>STQ</td>
<td>RESTORE ORIGINAL MQ.</td>
</tr>
<tr>
<td>00675 0560 00 0 00677</td>
<td>LDQ</td>
<td>RESTORE ORIGINAL MQ.</td>
</tr>
<tr>
<td>00676 0020 00 0 00001</td>
<td>TRA</td>
<td>SELECT TRAP RETURN.</td>
</tr>
<tr>
<td>00677 0 00000 0 00000</td>
<td>LSTM PZE</td>
<td>MQ STORAGE.</td>
</tr>
<tr>
<td>00700 0 00000 0 00000</td>
<td>LSTM</td>
<td>QP STORAGE.</td>
</tr>
<tr>
<td>00701 0021 00 0 00673</td>
<td>TLSTM</td>
<td>SELECT TRAP RETURN.</td>
</tr>
</tbody>
</table>
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 4, WIND UP THIS RUN.

<table>
<thead>
<tr>
<th>TTL</th>
<th>SECTION 4, WIND UP THIS RUN.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00702 -0760 00 0 00007</td>
<td>BACK LTM JUST IN CASE.</td>
</tr>
<tr>
<td>00703 0600 00 0 00005</td>
<td>STZ 5 RESET INTERVAL TIMER.</td>
</tr>
<tr>
<td>00704 -0061 00 0 00706</td>
<td>TCNT ++2 IF TAPE TIMER IN USE.</td>
</tr>
<tr>
<td>00705 -0340 00 0 00261</td>
<td>RCHT QUIT RESET TAPE TIMER.</td>
</tr>
<tr>
<td>00706 0074 00 4 00663</td>
<td>TSX LST,4 RESET TRAP, JUST IN CASE.</td>
</tr>
<tr>
<td>00707 -0520 00 0 00235</td>
<td>NZT TIT IF TAPE TIMERS USED.</td>
</tr>
<tr>
<td>00710 0772 00 0 02223</td>
<td>RMT Rewind TIMER TAPE.</td>
</tr>
<tr>
<td>00711 0520 00 0 01026</td>
<td>ZET EPMR TEST FOR ERROR PCS1 SPACE.</td>
</tr>
<tr>
<td>00712 0020 00 0 01401</td>
<td>TRA PNR YES, EXIT.</td>
</tr>
<tr>
<td>00713 0020 00 0 01403</td>
<td>TRA PNR+2 POST MORTEM IN ANY CASE.</td>
</tr>
</tbody>
</table>

SECTION TO GIVE FINAL EXIT TO MONITOR.

<table>
<thead>
<tr>
<th>TTL</th>
<th>SECTION 4, WIND UP THIS RUN.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00714 0074 00 4 02407</td>
<td>FINIS TSX WOT,4 END OF RUN COMMENT.</td>
</tr>
<tr>
<td>00715 0 00002 1 01027</td>
<td>PZE ENDRN,1,2 **</td>
</tr>
<tr>
<td>00716 0760 00 0 00004</td>
<td>ENK CHECK CONSOLE KEYS.</td>
</tr>
<tr>
<td>00717 -0130 00 0 00000</td>
<td>XCL **</td>
</tr>
<tr>
<td>00720 0044 00 0 00000</td>
<td>PNL **</td>
</tr>
<tr>
<td>00721 -0056 00 040000</td>
<td>LNT 040000 IF KEY 3 IS UP,</td>
</tr>
<tr>
<td>00722 0020 00 0 0765</td>
<td>TRA DOOR DONT CHECK KEYS AT ALL.</td>
</tr>
</tbody>
</table>

STOP RNT 2 CHECK FOR STOP INSTRUCTION.

<table>
<thead>
<tr>
<th>TTL</th>
<th>SECTION 4, WIND UP THIS RUN.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00723 0056 00 0 00002</td>
<td>STOP RNT 2 CHECK FOR STOP INSTRUCTION.</td>
</tr>
<tr>
<td>00724 0020 00 0 07732</td>
<td>TRA FILE NO, CHECK FOR REQUEST FOR EOF ON A3.</td>
</tr>
<tr>
<td>00725 0074 00 4 02520</td>
<td>TSX PRINT,4 YES, COMMENT.</td>
</tr>
<tr>
<td>00726 -0 00014 1 01031</td>
<td>MZE XSTOP,0,12 **</td>
</tr>
<tr>
<td>00727 0760 00 0 00004</td>
<td>ENK GET THE KEYS.</td>
</tr>
<tr>
<td>00730 -0130 00 0 00000</td>
<td>XCL PLACE IN AC.</td>
</tr>
<tr>
<td>00731 0044 00 0 00000</td>
<td>PAI PLACE IN SI.</td>
</tr>
</tbody>
</table>

FILE RNT 4 CHECK FOR EOF ON A3 REQUEST.

<table>
<thead>
<tr>
<th>TTL</th>
<th>SECTION 4, WIND UP THIS RUN.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00732 0056 00 0 00004</td>
<td>FILE RNT 4 CHECK FOR EOF ON A3 REQUEST.</td>
</tr>
<tr>
<td>00733 0020 00 0 07742</td>
<td>TRA CLOSE NO, CHECK FOR CLOSE TAPE INSTRUCTION.</td>
</tr>
<tr>
<td>00734 0770 00 0 01203</td>
<td>WEA 3 YES, WRITE EOF ON A3.</td>
</tr>
<tr>
<td>00735 0074 00 4 02520</td>
<td>TSX PRINT,4 COMMENT.</td>
</tr>
<tr>
<td>00736 -0 00014 0 01045</td>
<td>MZE XFILE,0,12 **</td>
</tr>
<tr>
<td>00737 0760 00 0 00004</td>
<td>ENK GET THE KEYS.</td>
</tr>
<tr>
<td>00740 -0130 00 0 00000</td>
<td>XCL PLACE IN AC.</td>
</tr>
<tr>
<td>00741 0044 00 0 00000</td>
<td>PAI PLACE IN SI.</td>
</tr>
</tbody>
</table>

CLOSE RNT 1 CHECK FOR CLOSE OUTPUT TAPE INSTRUCTION.

<table>
<thead>
<tr>
<th>TTL</th>
<th>SECTION 4, WIND UP THIS RUN.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00742 0056 00 0 00001</td>
<td>CLOSE RNT 1 CHECK FOR CLOSE OUTPUT TAPE INSTRUCTION.</td>
</tr>
<tr>
<td>00743 0020 00 0 07765</td>
<td>TRA DOOR NO, GO TO DOOR.</td>
</tr>
<tr>
<td>00744 0774 00 1 00012</td>
<td>AXT 10,11 SET TO COMMENT 10 TIMES.</td>
</tr>
<tr>
<td>00745 0074 00 4 02607</td>
<td>TSX WOT,4 YES, WRITE (END OF TAPE) AND</td>
</tr>
<tr>
<td>00746 0 00003 0 01077</td>
<td>PZE ENCTP,0,3 **</td>
</tr>
<tr>
<td>00747 0074 00 4 02407</td>
<td>TSX WOT,4 SKIP TO CHANNEL FOUR.</td>
</tr>
<tr>
<td>00750 0 00001 0 03147</td>
<td>PZE =H4 0,1 **</td>
</tr>
<tr>
<td>00751 0770 00 0 01203</td>
<td>WEA 3 THEN AN EOF MARK.</td>
</tr>
<tr>
<td>00752 0 00011 0 01203</td>
<td>TIX =-5,1,1 **</td>
</tr>
<tr>
<td>00753 0074 00 4 02407</td>
<td>TSX WOT,4 COMMENT, ITS REALLY THE END.</td>
</tr>
<tr>
<td>00754 0 00011 0 01102</td>
<td>PZE RENDTP,0,9 **</td>
</tr>
<tr>
<td>00755 0074 00 4 02407</td>
<td>TSX WOT,4 SKIP TO CHANNEL FOUR.</td>
</tr>
<tr>
<td>00756 0 00001 0 03147</td>
<td>PZE =H4 0,1 **</td>
</tr>
<tr>
<td>00757 0774 00 4 00012</td>
<td>AXT 10,4 TEN EOF'S ON A3.</td>
</tr>
<tr>
<td>00760 0770 00 0 01203</td>
<td>WEA 3 **</td>
</tr>
</tbody>
</table>

II7
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 4, WIND UP THIS RUN.

00761 2 00001 4 00760 TIX *=1,4,1**
00762 -0772 00 0 01203 RUNA 3 THEN RUN OUTPUT TAPE.
00763 0074 00 4 02520 TIX PRINT,4 COMMENT.
00764 -0 0016 0 01061 MZE XCLOSE,0,14**

00765 0600 00 0 03125 DOOR STZ *=0 MAKE A ZERO LOCATION.
00766 0564 00 0 03125 ENB *=0 RESET THE TRAPS.
00767 -0760 00 0 00007 LTM JUST IN CASE.
00770 0074 00 4 00663 TSX LSTM,4**
00771 -0760 00 0 00010 LSNM ** GET CONSOLE KEYS.
00772 0760 00 0 00004 ENK PLACE IN AC.
00773 -0130 00 0 00000 XCL PLACE IN INDICATORS.
00774 0444 00 0 00000 PAI PLACE IN INDICATORS.
00775 -0056 00 0 00001 LNT 1 CHECK IF OPERATOR WANTS TO STOP.
00776 0020 00 0 01002 TRA DOOR1 NO.
00777 0074 00 4 02520 TSX PRINT,4 YES, COMMENT.
01000 -0 0013 0 01113 MZE SEXIT,0,11**
01001 0020 00 0 00765 TRA DOOR GO BACK TO CHECK KEYS.

01002 0022 00 0 01003 DOOR1 TRCA **1 TURN OFF RC LIGHT.
01003 0774 00 1 00005 AXT 5,1 SET UP FOR 5 TRIES.
01004 0772 00 0 01201 REWA 1 REWIND THE SYSTEMS TAPE.
01005 0762 00 0 01221 RTBA 1 SELECT.
01006 0540 00 0 01024 RCHM LOAD LOAD SEQUENCE OF CHANNEL COMMANDS.
01007 -0054 00 0 00002 LFT 2 MISTART OR EXIT.
01010 0020 00 0 01013 TRA **3 MISTART.
01011 0762 00 0 01221 RTBA 1 EXIT, SKIP TWO RECORDS.
01012 0762 00 0 01221 RTBA 1**
01013 0460 00 0 01013 TCOA ** WAIT FOR OSCA.
01014 0022 00 0 01020 TRCA **4 EXIT FOR READ ERROR.
01015 -0500 00 0 01015 CAL 9 SET PREFIX OF 34 TO MZE.
01016 0630 00 0 00042 STP 34**
01017 0020 00 0 00001 TRA 1 THEN RETURN TO MONITOR.
01020 2 00001 1 01004 TIX DOOR1+2,1,1 READING ERROR, INDEX AND TRY AGAIN.
01021 0074 00 4 02520 TSX PRINT,4 FIVE REDUNDANCY FAILURES, COMMENT.
01022 -0 00017 0 01126 MZE XRCA(E,0,15**) CHANNEL COMMANDS.
01023 0020 00 0 00765 TRA DOOR GO TRY FIVE MORE TIMES.
01024 -0 00003 0 0000 LOAD IOCP 0,0,3**
01025 1 0000 0 0000 TCH 0**
01026 0 0000 0 0000 EPMR PZE TEST CELL FOR ERROR POST MORTEM.
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 4, WIND UP THIS RUN.

EJECT
TITLE SUPPRESS GENERATED OCTAL LISTING.
COMMENTS.

01027 002545246046 ENDRN BCI 2, END OF RUN.
01031 004225706003 XSTOP BCI 9, OKEY 34 DOWN, STOP COMMAND WHILE CHECKING KEYS, PRESS
01042 626321516360 BCI 3, START TO CONTINUE.
01045 004225706003 XFILE BCI 9, OKEY 33 DOWN, AN EOD HAS BEEN WRITTEN ON A3, PRESS STA
01056 516360634660 BCI 3, RT TO CONTINUE.
01061 004225706003 XCLOSE BCI 9, OKEY 35 DOWN, TAPE A3 HAS BEEN CLOSED, CHANGE TAPE AND
01072 604751256262 BCI 5, PRESS START TO CONTINUE.
01077 012545246046 ENOTP BCI 3, IEND OF TAPE.
01102 006330255125 RENDTP BCI 9, THERE REALLY ISN'T ANY MORE ON THIS TAPE - HONEST.
01113 004225706001 SEXIT BCI 9, OKEY 17 DOWN, STOP COMMAND BEFORE EXIT, PRESS START TO
01124 602364645631 BCI 2, CONTINUE.
01126 002631652560 XRCAL BCI 9, FIVE CONSECUTIVE REDUNDANCY FAILURES IN READING A1. P
01137 512562626062 BCI 6, RESS START FOR FIVE MORE TRIES.

DETAIL RETURN TO NORMAL LISTING MODE.

TT140970
TT140980
TT140990
TT141000
TT141010
TT141020
TT141030
TT141040
TT141050
TT141060
TT141070
TT141080
TT141090
TT141100
TT141110
TT141120
TT141130
TT141140
TT141150
TT141160
TT141170
TT141180
TT141190
TT141200
TT141210
TT141220
TT141230
TT141240
TT141250
TT141260
TT141270
TT141280
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 5, SVCON, PMR, OCT, OCTADR, AND TABBLK.

SPACE 2
CONSOLE LIGHTS SAVING ROUTINE.

SPACE 2
ACCESS TO OCTAL CONVERTER, DELETE LEADING ZEROS.

SPACE 2
TABLE FOR DELETING LEADING ZEROS.
<table>
<thead>
<tr>
<th>Location 1</th>
<th>Instruction 1</th>
<th>Description 1</th>
<th>Location 2</th>
<th>Instruction 2</th>
<th>Description 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>01212</td>
<td>030000001221</td>
<td>VFD H6/3,15/0,15/TABBLK+1C</td>
<td>01213</td>
<td>040000001221</td>
<td>H6/4,15/0,15/TABBLK+10</td>
</tr>
<tr>
<td>01214</td>
<td>050000001221</td>
<td>H6/5,15/0,15/TABBLK+10</td>
<td>01215</td>
<td>060000001221</td>
<td>H6/6,15/0,15/TABBLK+10</td>
</tr>
<tr>
<td>01216</td>
<td>070000001221</td>
<td>H6/7,15/0,15/TABBLK+10</td>
<td>01217</td>
<td>100000001221</td>
<td>H6/8,15/0,15/TABBLK+10</td>
</tr>
<tr>
<td>01220</td>
<td>110000001221</td>
<td>H6/9,15/0,15/TABBLK+10</td>
<td>01221</td>
<td>000000001221</td>
<td>H6/10,15/0,15/TABBLK+10</td>
</tr>
<tr>
<td>01222</td>
<td>001000001221</td>
<td>H6/11,15/0,15/TABBLK+1C</td>
<td>01223</td>
<td>020000001221</td>
<td>H6/12,15/0,15/TABBLK+1C</td>
</tr>
<tr>
<td>01224</td>
<td>030000001221</td>
<td>H6/13,15/0,15/TABBLK+1C</td>
<td>01225</td>
<td>040000001221</td>
<td>H6/14,15/0,15/TABBLK+1C</td>
</tr>
<tr>
<td>01226</td>
<td>050000001221</td>
<td>H6/15,15/0,15/TABBLK+1C</td>
<td>01227</td>
<td>060000001221</td>
<td>H6/16,15/0,15/TABBLK+1C</td>
</tr>
<tr>
<td>01230</td>
<td>070000001221</td>
<td>H6/17,15/0,15/TABBLK+1C</td>
<td>01231</td>
<td>100000001221</td>
<td>H6/18,15/0,15/TABBLK+1C</td>
</tr>
<tr>
<td>01232</td>
<td>110000001221</td>
<td>H6/19,15/0,15/TABBLK+10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SPACE 2**

**COMPLEMENT ADDRESS TO OCTAL-BCI CONVERTER.**

<table>
<thead>
<tr>
<th>Location 1</th>
<th>Instruction 1</th>
<th>Description 1</th>
<th>Location 2</th>
<th>Instruction 2</th>
<th>Description 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>01233</td>
<td>0634 00 4 01256</td>
<td>COMADR SXA COMRT,4 SAVE IR4.</td>
<td>01234</td>
<td>0774 00 4 01251</td>
<td>AXT COMSW+1,4 RESET SWITCH.</td>
</tr>
<tr>
<td>01235</td>
<td>0634 00 4 01250</td>
<td>SXA COMSW+4 <strong>+</strong></td>
<td>01236</td>
<td>-0130 00 0 00000</td>
<td>XCL <strong>+</strong></td>
</tr>
<tr>
<td>01237</td>
<td>-0100 00 0 01242</td>
<td>TNZ <strong>+</strong></td>
<td>01240</td>
<td>0560 00 0 03157</td>
<td>LDQ <strong>-</strong></td>
</tr>
<tr>
<td>01241</td>
<td>0020 00 0 00000</td>
<td>TRA COMRT AND EXIT.</td>
<td>01242</td>
<td>0737 00 4 00000</td>
<td>PAC 0,4</td>
</tr>
<tr>
<td>01243</td>
<td>0754 00 4 00000</td>
<td>PIA 0,4</td>
<td>01244</td>
<td>-0765 00 0 00017</td>
<td>LGT 15</td>
</tr>
<tr>
<td>01245</td>
<td>0774 00 4 00005</td>
<td>AXT 5,4</td>
<td>01246</td>
<td>0767 00 0 00003</td>
<td>ALS 3</td>
</tr>
<tr>
<td>01247</td>
<td>-0763 00 0 00003</td>
<td>LGL 3</td>
<td>01250</td>
<td>0020 00 0 01251</td>
<td>COMSW+1 SWITCH AFTER FIRST NON-ZERO DIGIT.</td>
</tr>
<tr>
<td>01251</td>
<td>0100 00 0 01254</td>
<td>TRA <strong>+</strong></td>
<td>01252</td>
<td>-0501 00 0 03157</td>
<td>TZE <strong>+</strong></td>
</tr>
<tr>
<td>01253</td>
<td>-0625 00 0 01250</td>
<td>ORA <strong>-</strong></td>
<td>01254</td>
<td>2 0001 4 01246</td>
<td>STL COMSW+1 SET SWITCH.</td>
</tr>
<tr>
<td>01255</td>
<td>-0130 00 0 00000</td>
<td>XCL <strong>+</strong></td>
<td>01256</td>
<td>0774 00 4 00000</td>
<td>COMRT AXT <strong>+</strong></td>
</tr>
<tr>
<td>01257</td>
<td>0020 00 4 00001</td>
<td>TRA 1,4 RETURN.</td>
<td>01258</td>
<td>0634 00 4 01276</td>
<td>OCT SXA <strong>+14,4</strong></td>
</tr>
<tr>
<td>01259</td>
<td>0774 00 4 00006</td>
<td>AXT 6,4</td>
<td>01260</td>
<td>-0754 00 0 00000</td>
<td>PXO 0,0</td>
</tr>
<tr>
<td>01261</td>
<td>0767 00 0 00003</td>
<td>ALS 3</td>
<td>01262</td>
<td>-0763 00 0 00003</td>
<td>LGT 3</td>
</tr>
<tr>
<td>01263</td>
<td>0001 4 01263</td>
<td>TIX <strong>2,-4,1</strong></td>
<td>01264</td>
<td>0602 00 0 01300</td>
<td>SWL <strong>+10</strong></td>
</tr>
<tr>
<td>Address</td>
<td>Code</td>
<td>Instruction</td>
<td>Comment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>-------------</td>
<td>---------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01267</td>
<td>0774 00 4 00006</td>
<td>AXT 6,4</td>
<td>CONVERT RIGHT HALF.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01270</td>
<td>0754 00 0 00000</td>
<td>PXD 0,0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01271</td>
<td>0767 00 0 00003</td>
<td>ALS 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01272</td>
<td>0763 00 0 00003</td>
<td>LGL 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01273</td>
<td>00001 4 01271</td>
<td>TIX *2,4,1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01274</td>
<td>0130 00 0 00000</td>
<td>XCL</td>
<td>LOAD COMPLETED WORD.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01275</td>
<td>0500 00 0 01300</td>
<td>CAL **3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01276</td>
<td>0774 00 4 00000</td>
<td>AXT **4</td>
<td>RESTORE IR4.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01277</td>
<td>0020 00 4 00001</td>
<td>TRA 1,4</td>
<td>RETURN TO CALLER.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01300</td>
<td>0 00000 0 00000</td>
<td>PZE</td>
<td>TEMPORARY STORAGE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01260</td>
<td>OCT</td>
<td>SYN OCT</td>
<td>EXTERNAL NAME FOR OCT.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SPACE 2**

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>01301</td>
<td>000000001301</td>
<td>BCDTAB VFD</td>
<td>H6/0,15/0,15/BCDOTAB 00</td>
</tr>
<tr>
<td>01302</td>
<td>01000001301</td>
<td>VFD</td>
<td>H6/1,15/0,15/BCDOTAB 01</td>
</tr>
<tr>
<td>01303</td>
<td>02000001301</td>
<td>VFD</td>
<td>H6/2,15/0,15/BCDOTAB 02</td>
</tr>
<tr>
<td>01304</td>
<td>03000001301</td>
<td>VFD</td>
<td>H6/3,15/0,15/BCDOTAB 03</td>
</tr>
<tr>
<td>01305</td>
<td>04000001301</td>
<td>VFD</td>
<td>H6/4,15/0,15/BCDOTAB 04</td>
</tr>
<tr>
<td>01306</td>
<td>05000001301</td>
<td>VFD</td>
<td>H6/5,15/0,15/BCDOTAB 05</td>
</tr>
<tr>
<td>01307</td>
<td>06000001301</td>
<td>VFD</td>
<td>H6/6,15/0,15/BCDOTAB 06</td>
</tr>
<tr>
<td>01310</td>
<td>07000001301</td>
<td>VFD</td>
<td>H6/7,15/0,15/BCDOTAB 07</td>
</tr>
<tr>
<td>01311</td>
<td>10000001301</td>
<td>VFD</td>
<td>H6/8,15/0,15/BCDOTAB 10</td>
</tr>
<tr>
<td>01312</td>
<td>11000001301</td>
<td>VFD</td>
<td>H6/9,15/0,15/BCDOTAB 11</td>
</tr>
<tr>
<td>01313</td>
<td>54000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 12</td>
</tr>
<tr>
<td>01314</td>
<td>13000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 13</td>
</tr>
<tr>
<td>01315</td>
<td>14000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 14</td>
</tr>
<tr>
<td>01316</td>
<td>54000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 15</td>
</tr>
<tr>
<td>01317</td>
<td>54000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 16</td>
</tr>
<tr>
<td>01320</td>
<td>54000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 17</td>
</tr>
<tr>
<td>01321</td>
<td>20000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 19</td>
</tr>
<tr>
<td>01322</td>
<td>21000001301</td>
<td>VFD</td>
<td>H6/A,15/0,15/BCDOTAB 21</td>
</tr>
<tr>
<td>01323</td>
<td>22000001301</td>
<td>VFD</td>
<td>H6/B,15/0,15/BCDOTAB 22</td>
</tr>
<tr>
<td>01324</td>
<td>23000001301</td>
<td>VFD</td>
<td>H6/C,15/0,15/BCDOTAB 23</td>
</tr>
<tr>
<td>01325</td>
<td>24000001301</td>
<td>VFD</td>
<td>H6/D,15/0,15/BCDOTAB 24</td>
</tr>
<tr>
<td>01326</td>
<td>25000001301</td>
<td>VFD</td>
<td>H6/E,15/0,15/BCDOTAB 25</td>
</tr>
<tr>
<td>01327</td>
<td>26000001301</td>
<td>VFD</td>
<td>H6/F,15/0,15/BCDOTAB 26</td>
</tr>
<tr>
<td>01330</td>
<td>27000001301</td>
<td>VFD</td>
<td>H6/H,15/0,15/BCDOTAB 27</td>
</tr>
<tr>
<td>01331</td>
<td>30000001301</td>
<td>VFD</td>
<td>H6/H,15/0,15/BCDOTAB 30</td>
</tr>
<tr>
<td>01332</td>
<td>31000001301</td>
<td>VFD</td>
<td>H6/I,15/0,15/BCDOTAB 31</td>
</tr>
<tr>
<td>01333</td>
<td>54000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 32</td>
</tr>
<tr>
<td>01334</td>
<td>33000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 33</td>
</tr>
<tr>
<td>01335</td>
<td>34000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 34</td>
</tr>
<tr>
<td>01336</td>
<td>54000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 35</td>
</tr>
<tr>
<td>01337</td>
<td>54000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 36</td>
</tr>
<tr>
<td>01340</td>
<td>54000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 37</td>
</tr>
<tr>
<td>01341</td>
<td>40000001301</td>
<td>VFD</td>
<td>H6/**,15/0,15/BCDOTAB 40</td>
</tr>
<tr>
<td>01342</td>
<td>41000001301</td>
<td>VFD</td>
<td>H6/J,15/0,15/BCDOTAB 41</td>
</tr>
<tr>
<td>01343</td>
<td>42000001301</td>
<td>VFD</td>
<td>H6/K,15/0,15/BCDOTAB 42</td>
</tr>
<tr>
<td>01344</td>
<td>43000001301</td>
<td>VFD</td>
<td>H6/L,15/0,15/BCDOTAB 43</td>
</tr>
<tr>
<td>01345</td>
<td>44000001301</td>
<td>VFD</td>
<td>H6/M,15/0,15/BCDOTAB 44</td>
</tr>
<tr>
<td>01346</td>
<td>45000001301</td>
<td>VFD</td>
<td>H6/N,15/0,15/BCDOTAB 45</td>
</tr>
<tr>
<td>OFFSET</td>
<td>ADDRESS</td>
<td>VFD</td>
<td>SDM</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>01347</td>
<td>460000001301</td>
<td>H6/0,15/0,15/BCDOTAB</td>
<td>46</td>
</tr>
<tr>
<td>01350</td>
<td>470000001301</td>
<td>H6/P,15/0,15/BCDOTAB</td>
<td>47</td>
</tr>
<tr>
<td>01351</td>
<td>500000001301</td>
<td>H6/Q,15/0,15/BCDOTAB</td>
<td>50</td>
</tr>
<tr>
<td>01352</td>
<td>510000001301</td>
<td>H6/R,15/0,15/BCDOTAB</td>
<td>51</td>
</tr>
<tr>
<td>01353</td>
<td>540000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>52</td>
</tr>
<tr>
<td>01354</td>
<td>530000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>53</td>
</tr>
<tr>
<td>01355</td>
<td>540000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>54</td>
</tr>
<tr>
<td>01356</td>
<td>540000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>55</td>
</tr>
<tr>
<td>01357</td>
<td>540000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>56</td>
</tr>
<tr>
<td>01360</td>
<td>540000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>57</td>
</tr>
<tr>
<td>01361</td>
<td>600000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>58</td>
</tr>
<tr>
<td>01362</td>
<td>610000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>59</td>
</tr>
<tr>
<td>01363</td>
<td>620000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>60</td>
</tr>
<tr>
<td>01364</td>
<td>630000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>61</td>
</tr>
<tr>
<td>01365</td>
<td>640000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>62</td>
</tr>
<tr>
<td>01366</td>
<td>650000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>63</td>
</tr>
<tr>
<td>01367</td>
<td>660000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>64</td>
</tr>
<tr>
<td>01370</td>
<td>670000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>65</td>
</tr>
<tr>
<td>01371</td>
<td>700000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>66</td>
</tr>
<tr>
<td>01372</td>
<td>710000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>67</td>
</tr>
<tr>
<td>01373</td>
<td>740000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>68</td>
</tr>
<tr>
<td>01374</td>
<td>730000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>69</td>
</tr>
<tr>
<td>01375</td>
<td>740000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>70</td>
</tr>
<tr>
<td>01376</td>
<td>740000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>71</td>
</tr>
<tr>
<td>01377</td>
<td>740000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>72</td>
</tr>
<tr>
<td>01400</td>
<td>540000001301</td>
<td>H6/+,15/0,15/BCDOTAB</td>
<td>73</td>
</tr>
</tbody>
</table>

Note: No Blanks Allowed in VFD.

Note: No Commas Allowed in VFD.
TESTS1 FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 5, SVCON, PMR, CNT, OCTDR, AND TABBK.

01430 0560 00 0 03142  LDQ =HOOCAL
01431 0500 00 0 02022  CLA =PMCN
01432 0402 00 0 03131  SUB =5
01433 0120 0 0 03137  TPL =4
01434 0560 00 0 03145  LDQ =HOOCAL
01435 0500 00 0 03135  CLA =59
01436 0020 0 0 01440  TRA =2
01437 0400 0 0 03130  ADD =3
01438 0601 0 0 02022  STD =PMCN
01439 0600 0 0 01503  STQ =PM
01440 0740 0 0 02407  TSX =W0T,4
01441 0 0005 0 01503  PZE =PM:+0,5
01442 0074 0 0 01640  TSX =DG0M,4
01443 0 0000 0 00000  PZE =**0,00**
01444 0 0000 0 01447  TXI =**1,12,-1
01445 1 7777 2 01447  COD1 PZE
01446 3 0000 0 01413  LSUB TXH =SLPM+1,2,**
01447 3 0000 0 01413  COUNT INSERTED BY MAPP, NEXT POST MORTEM.

01450 0560 0 0 03142  LDQ =HOOCAL
01451 0500 0 0 02022  CLA =PMCN
01452 0402 0 0 03131  SUB =5
01453 0120 0 0 03147  TPL =4
01454 0560 0 0 03145  LDQ =HOOCAL
01455 0500 0 0 03135  CLA =59
01456 0020 0 0 01460  TRA =2
01457 0400 0 0 03130  ADD =3
01458 0601 0 0 02022  STD =PMCN
01459 0600 0 0 01503  STQ =PM
01460 0740 0 0 02407  TSX =W0T,4
01461 0 0007 0 01510  PZE =PM:+0,7
01462 0050 0 0 00234  CAL =ORG
01463 0621 0 0 01467  STA =2
01464 0074 0 0 01640  TSX =DG0M,4
01465 0 0040 0 00000  PZE =**0,256**
01466 0074 0 0 02407  TSX =W0T,4
01467 0 0004 0 01517  PZE =ENDM+0,4
01468 0760 0 0 00140  SLF =ENDS
01469 0200 0 0 00714  TRA =FIN

01470 004622412523  PR0ER BCI 7,000 PROGRAM ERROR, PROGRAM TERMINATED.
01471 636047514627  7,000
01472 51214602551  OCTAL
01473 51465136047  ASSEMBLED
01474 51465121444  PROGRAM
01475 603635114431  TERMINATED.
01476 45216352433  PMH
01477 00462332143  BCI
01478 602464444670  5,000 OCTAL DUMP OF PMH CONVOLVES.
01479 602464444670  BCI
01480 602464444670  7,000 OCTAL DUMP OF ASSEMBLED PROGRAM.

PMH

PMBP

5,000 OCTAL DUMP OF PMH CONVOLVES.

01500 00462332143  PMBP
01501 602464444670  BCI
01502 602464444670  7,000 OCTAL DUMP OF ASSEMBLED PROGRAM.
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 5. SVCON, PMR, OCT, OCTADIR, AND TABBKL.

ENDPM 8CI 4.0END OF POST MORTEM.

TT152470

SPACE 2
USE RESULTS OF SVCON TO DUMP CONSOLE LIGHTS.

TT152480

TT152490

TT152500

TT152510

TT152520

TT152530

TT152540

TT152550

TT152560

TT152570

TT152580

01523 0634 00 0 01602
DCON SXA DCONX4,4
01524 0560 00 0 01172
LDQ SVPO S=Q,P.
01525 0754 00 0 00000
PXK 0,0 CLEAR AC.
01526 0763 00 0 00001
LGL 1 **
01527 0767 00 0 00013
ALS 11 **
01530 0763 00 0 00001
LGL 1 **
01531 0767 00 0 00013
ALS 11 **
01532 0763 00 0 00001
LGL 1 **
01533 0501 00 0 03155
DRA +H 0,0,0 PUT IN COMMAS.
01534 0602 00 0 01612
SLW HEAD02+2 INSERT.
01535 0560 00 0 01170
LDQ AC CONVERT AC TO OCTAL.
01536 0074 00 4 01260
TSX OCT4 **
01537 0602 00 0 01615
SLW HEAD03+2 INSERT.
01540 0600 00 0 01616
STQ HEAD03+3 **
01541 0560 00 0 01171
LDQ MQ CONVERT MQ TO OCTAL.
01542 0074 00 4 01260
TSX OCT4 **
01543 0602 00 0 01621
SLW HEAD04+2 INSERT.
01544 0600 00 0 01622
STQ HEAD04+3 **
01545 0560 00 0 01167
LDQ SIND CONVERT SI TO OCTAL.
01546 0074 00 4 01260
TSX OCT4 **
01547 0602 00 0 01625
SLW HEAD05+2 INSERT.
01550 0600 00 0 01626
STQ HEAD05+3 **
01551 0560 00 0 01164
LDQ IRI CONVERT IRI.
01552 0074 00 4 01173
TSX OCTADIR4 **
01553 0600 00 0 01631
STQ HEAD06+2 INSERT.
01554 0560 00 0 01165
LDQ IRI CONVERT IRI.
01555 0074 00 4 01173
TSX OCTADIR4 **
01556 0600 00 0 01634
STQ HEAD07+2 **
01557 0560 00 0 01166
LDQ IRI CONVERT IRI.
01560 0074 00 4 01173
TSX OCTADIR4 **
01561 0600 00 0 01637
STQ HEAD08+2 INSERT.
01562 0074 00 4 02407
TSX W0T,4 PRINT CONSOLE.
01563 00004 0 01604
PZE HEAD01,0,4 **
01564 0074 00 4 02407
TSX W0T,4 **
01565 0 00003 0 01610
PZE HEAD02,0,3 **
01566 0074 00 4 02407
TSX W0T,4 **
01567 0 00004 0 01613
PZE HEAD03,0,4 **
01570 0074 00 4 02407
TSX W0T,4 **
01571 0 00004 0 01617
PZE HEAD04,0,4 **
01572 0074 00 4 02407
TSX W0T,4 **
01573 0 00004 0 01623
PZE HEAD05,0,4 **
01574 0074 00 4 02407
TSX W0T,4 **
01575 0 00003 0 01627
PZE HEAD06,0,3 **
01576 0074 00 4 02407
TSX W0T,4 **
TEST3 FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 5, SVCON, PRM, OCT, OCTADR, AND TABBLK.

01577 000003 001632 PIE HEAD07, 0,3 **
01600 0074 00 402407 TSX W01, 4 **
01601 000003 001635 PIZ HEAD08, 0, 3 **
01602 0774 00 400000 CCXN4 AXT **, 4 RESTORE IR4.
01603 0020 00 400001 TRA 1, 4 RETURN.

01604 01476626360 HEAD01 BCI 4, 1POST MORTEM OF CONSOLE.
01605 044651632544 HEAD02 BCI 3, 0 S,Q,P = 0, 0, 0
01606 604626602346 HEAD03 BCI 4, C AC = 000000000000
01607 456246432533 HEAD04 BCI 4, 0 MQ = 000000000000
01610 006006060600 HEAD05 BCI 4, 0 SI = 000000000000
01611 627350734713 HEAD06 BCI 3, 0 IR1 = 000000
01612 600073007300 HEAD07 BCI 3, 0 IR2 = 000000
01613 006006060600 HEAD08 BCI 3, 0 IR4 = 000000
01614 212360136060 SPACE 2
01615 000000000000 OCTDMP, OCTAL DUMP WITH MNEMONICS.
01616 000000000000
01617 000000000000

SPACE 2
OCTDMP, OCTAL DUMP WITH MNEMONICS.

01640 0634 00 402016 OCTDMP SXA DX1, 4 SAVE IRS.
01641 0634 00 202017 SXA DX2, 2 **
01642 0634 00 102020 SXA DX1, 1 **
01643 0500 00 400001 CAL 1, 4 GET CALLING SEQUENCE.
01644 0734 00 100000 PDX 0, 1 COUNT TO IR1.
01645 0621 00 020233 STA ILC FIRST LOCATION.
01646 0771 00 00022 STA IRC FIRST + N = LAST.
01647 0361 00 400001 ACR IR8 ADR ERG 2 COMPUTE END ADDRESS.
01650 0621 00 01660 STA LOOP1 SET UP LOOP.
01651 0621 00 01664 STA LOOP1+4 **
01652 0621 00 01742 STA REG 2 **
01653 0560 00 020233 ADR LOQ ILC GET ILC AND CONVERT.
01654 0074 00 401173 TSX OCTADR, 4 **
01655 0773 00 000006 ROL 6 PUT BLANK AT END.
01656 0600 00 02746 STQ OUT1 INSERT.
01657 0774 00 200022 LOOP2 AXT 18, 2 SET WORD COUNT.
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 5. SVCON, PMR, OCT, OCTADR, AND TABBK.

01660 0500 0 1 00000 LOOP1 CAL ***.1 GET WORD.
01661 00017 2 01740 TXL REG,2,15 IF FIRST WORD, CHECK FOR REPEATS.
01662 0602 0 0 02024 SLW LWORD SAVE WORD.
01663 0774 0 0 00000 AXT 0,4 SET IRX.
01664 -0340 0 1 00000 LAS ***.1 CHECK FOR REPEATS.
01665 0020 0 0 01672 TRA ++5 NOT SAME, EXIT.
01666 1 00001 4 01670 TXI +2,4,1 SAME, INDEX.
01667 0020 0 0 01672 TRA ++3 NOT SAME, EXIT.
01668 2 00001 1 01664 TXI ++,1,1 INDEX AND TRY NEXT WORD.
01669 1 7777 4 01672 TXI +1,4,-1 LAST WORD, INDEX FOR LINE AT BOTTOM.
01670 3 00022 4 01675 TXH +3,4,1,8 IF MORE THAN 10 REPEATS, COMMENT.
01671 -0634 0 0 01674 XKD ++1,4 LESS THAN SIX, RETURN TO NORMAL.
01672 1 00000 1 01740 TXI REG,1,1 MORE THAN SIX, CONVERT NUMBER.
01673 0634 0 0 02025 SLQ NREP,4 PLACE IN MQ.
01700 0774 0 0 00000 STZ DNREP CLEAR DECIMAL NREP.
01701 0754 0 0 00000 PXD 0,0 SET CONVERSION LOOP.
01702 0221 0 0 03133 DVP =10 CLEAR AC.
01703 0767 0 0 00036 ALS 30,4 MOD TEN.
01704 -0602 0 0 02026 STQ DNREP SHIFT TO POSITION.
01705 2 00000 4 02101 TIX -4,4,6 INSERT.
01706 0560 0 0 02026 LDQ DNREP INDEX.
01707 -0154 0 0 01207 CRQ TABBK,0,6 DELETE LEADING ZEROS.
01710 -0600 0 0 02037 STQ WREP=8 PLACE IN COMMENT.
01711 -0500 0 0 02024 STQ LWORD CONVERT WORD.
01712 0074 0 0 02047 TXS (OPCD),4
01713 0602 0 0 02043 SLW WREP=12
01714 0560 0 0 02024 TXS OCT,4
01715 0074 0 0 01260 SLW WREP=13
01717 -0600 0 0 02045 STQ WREP=14 ASSUME DOUBLE SPACE FOR COMMENT.
01720 0560 0 0 03143 LDQ =MC GET POST MORTEN LINE COUNT.
01721 0500 0 0 02022 CLA PMCNT
01722 0402 0 0 03130 SUB =3 REPEAT COMMENT TAKES THREE LINES.
01723 0120 0 0 01726 TPL =+3 SKIP IF IT FITS.
01724 0560 0 0 03146 LDQ =H SET TO EJECT PAGE.
01725 0500 0 0 03134 CLA =S CLEAR PM LINE COUNT.
01726 0601 0 0 02022 STQ PMCNT SAVE LINE COUNT.
01727 -0600 0 0 02027 STQ WREP INSERT CARRIAGE CONTROL.
01730 0074 0 0 02407 TXS WOT,4 WOT REPEATS.
01731 0 0020 0 02027 TXS WRP=0,16 BLANK LINE.
01732 0074 0 0 02407 TXS WOT,4 INSERT.
01733 0 0001 0 03160 PZE *H =0,1 INCREMENT ILC BY NREP.
01734 -0500 0 0 02023 CAL ILC
01735 0361 0 0 02025 ACL NREP
01736 0621 0 0 02023 STA ILC
01737 0020 0 0 01653 TRA ADR RETURN TO CONTROL LOOP.
01740 0074 0 0 02047 REG TXS (OPCD),4 NORMAL PATH, CONVERT WORD.
01741 0602 0 0 02771 SLW OUT=20,2
01742 0560 0 0 01000 LDQ ++,1 CONVERT TO OCTAL.
01743 0074 0 0 01260 TXS OCT,4
01744 0602 0 0 02772 SLW OUT=21,2
01745 -0600 0 0 02773 STQ OUT=22,2
TEST # FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 5, SVCON, PNR, OCT, OCTADR, AND TABBLK.

01746 -2 00001 1 01776  TIX  FIN 1 1  COUNT WORDS CONVERTED.
01747  2 00003 2 01660  TXI  LOOP 1 2 3  COUNT WORDS THIS LINE.

01750  0560 00 0 03160  PPM  LDQ =H
01751  0500 00 0 02022  CLA  PMCN'T
01752  0402 00 0 03126  SUB =1
01753  0120 00 0 01756  TPL **3
01754  0560 00 0 03146  LDQ =H1
01755  0500 00 0 03135  CLA =99
01756  0601 00 0 02022  STO  PMCN'T
01757  0600 00 0 02745  STQ  OUT
01760  0074 00 4 02407  TSX  WDT 4
01761  0 00024 0 02745  PZE  OUT 0 1 20
01762  0 00000 0 02746  CALL OUT 1
01763  -0320 00 0 03150  ANA =00707070777
01764  0361 00 0 03144  ACL =007070707600
01765  -0320 00 0 03150  ANA =00707070777
01766  -0130 00 0 00000  XCL
01767  -0134 04 0 01207  CRQ TABBLK 0 4
01770  -0773 00 0 00014  RQL 1 2
01771  -0600 00 0 02746  STQ OUT 1
01772  -0500 00 0 02023  CAL ILC
01773  0361 00 0 03132  ACL =6
01774  0621 00 0 02023  STA ILC
01775  0020 00 0 01697  TRA LOOP 2 FIN TIX **6 2 1 3
01776  -2 00003 2 02004  CAL =H
01777  -0500 00 0 03160  SLW OUT 20 2
01780  0602 00 2 02771  SLW OUT 22 2
02001  0602 00 2 02772  SLW OUT 21 2
02002  0602 00 2 02773  SLW OUT 22 2
02003  2 00003 2 02000  TIX =-3 2 1 3
02004  0560 00 0 03160  LDQ =H
02005  0500 00 0 02022  CLA PMCN'T
02006  0402 00 0 03126  SUB =1
02007  0120 00 0 02012  TPL **3
02010  0560 00 0 03146  LDQ =H1
02011  0500 00 0 03135  CLA =99
02012  0601 00 0 02022  STO PMCN'T
02013  -0600 00 0 02745  STQ OUT
02014  0074 00 4 02407  TSX WDT 4
02015  0 00024 0 02745  PZE OUT 0 20
02016  0 0774 00 4 00000  NX  AX T **4
02017  0774 00 2 00000  NX  AX T **2
02020  0774 00 1 00000  NX  AX T **1
02021  02020 00 4 00002  TRA 2 4
02022  0 00000 0 00055  PMCN'T PZE 45
02023  0 00000 0 00000  ILC PZE
02024  0 00000 0 00000  LWRC PZE
02025  0 00000 0 00000  NREP PZE
02026  0 00000 0 00000  CNREP PZE
02027  000000000000  WREP BCI 9 0
02030  606060606060
02031  606060606060
02032  606060606060
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 5, SVCON, PMR, GCT, OCTADDR, AND TABBLK.

BCI 7 CELLS ALL CONTAIN 0000000000000000

SPACE 2 (OPCD), FIND MNEMONIC OPERATION CODE.

0507 0634 00 2 02131 (OPCD) S$A AR2,2 SAVE IR2.
0508 0634 00 1 02132 S$A AR1,1 SAVE IR1.
0509 0774 00 1 00000 AXT 0,1 SET IR1.
0510 -0320 00 0 03165 ANA #077700000000 MASK OPCD.
0511 0100 00 0 02125 TIE OPFND HTR EXIT.
0512 0602 00 0 02135 SLW OPBIN SAVE OPCD.
0513 0630 00 0 02134 STP PRE SAVE PREFIX.
0514 0520 00 0 02134 ZET PRE IF NON-ZERO, TYPE A.
0515 0020 00 0 02117 TRA TYPEA ...
0516 0534 00 2 02406 LXP SIZE,2 SET IR2.
0517 0020 00 2 02116 TRA LOWER,2 BEGIN SEARCH.

0518 -3 77540 1 02066 SRC1 TXL SRC2,1,-NUM CHECK RANGE.
0519 -5000 00 1 02146 CAL TABL,1 GET TABLE ENTRY.
0520 -0320 00 0 03165 ANA #077700000000 MASK OPCD.
0521 0340 00 0 02135 LAS OPBIN COMPARE WITH OPBIN.
0522 1 7776 2 02115 SRC2 TXI RAISE,2,-2 BIGGER, GO TO RAISE INDEX.
0523 0020 00 0 02125 TRA OPFND FOUND, EXIT WITH INDEX.
0524 1 7776 2 02116 TXI LOWER,2,-2 SMALLER, GO TO LOWER INDEX.
0525 1 0040 1 02062 TXI SRCH1,1,1,256 TABLE, POWER OF TWO INCREMENTS.
0526 1 7740 1 02062 TXI SRCH1,1,1,256 **
0527 1 0020 01 02062 TXI SRCH1,1,1,128 **
0528 1 7760 1 02062 TXI SRCH1,1,1,128 **
0529 1 0010 1 02062 TXI SRCH1,1,1,64 **
0530 1 7700 1 02062 TXI SRCH1,1,1,64 **
0531 1 0004 1 02062 TXI SRCH1,1,1,32 **
0532 1 7740 1 02062 TXI SRCH1,1,1,32 **
0533 1 0002 1 02062 TXI SRCH1,1,1,16 **
0534 1 7760 1 02062 TXI SRCH1,1,1,16 **
0535 1 0010 1 02062 TXI SRCH1,1,1,8 **
0536 1 7770 1 02062 TXI SRCH1,1,1,8 **
0537 1 0004 1 02062 TXI SRCH1,1,1,4 **
0538 1 7774 1 02062 TXI SRCH1,1,1,4 **
0539 1 0002 1 02062 TXI SRCH1,1,1,2 **
0540 1 7776 1 02062 TXI SRCH1,1,1,2 **
0541 1 0001 1 02062 TXI SRCH1,1,1,1 **
0542 1 7777 1 02062 TXI SRCH1,1,1,1 **
0543 0020 00 0 02117 TRA TYPEA NOT FOUND, TYPE A.
### TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.

**SECTION 5, SYCON, PRK, GCT, OCTADR, AND TABBLK.**

<table>
<thead>
<tr>
<th>O2114</th>
<th>0020 00 0 02117</th>
<th>TRA</th>
<th>TYPEA</th>
<th>**</th>
<th>TT154770</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2115</td>
<td>0522 00 2 02115</td>
<td>RAISE XEC</td>
<td>*;2</td>
<td>RAISE INDEX.</td>
<td>TT154780</td>
</tr>
<tr>
<td>O2116</td>
<td>0522 00 2 02116</td>
<td>LOWER XEC</td>
<td>*;2</td>
<td>TT154790</td>
<td></td>
</tr>
<tr>
<td>C2117</td>
<td>0560 00 0 02135</td>
<td>TYPEA LDQ</td>
<td>OPBIN</td>
<td>TYPE A, PICKUP MNEMONIC.</td>
<td>TT154800</td>
</tr>
<tr>
<td>C2120</td>
<td>0754 00 0 00000</td>
<td>PXD</td>
<td>0,0</td>
<td>**</td>
<td>TT154820</td>
</tr>
<tr>
<td>C2121</td>
<td>0763 00 0 00033</td>
<td>LGL</td>
<td>3</td>
<td>**</td>
<td>TT154840</td>
</tr>
<tr>
<td>C2122</td>
<td>0734 00 1 00000</td>
<td>PAK</td>
<td>0,1</td>
<td>**</td>
<td>TT154850</td>
</tr>
<tr>
<td>C2123</td>
<td>0500 00 1 02145</td>
<td>CAL</td>
<td>LSTA+7,1</td>
<td>PICKUP MNEMONIC.</td>
<td>TT154860</td>
</tr>
<tr>
<td>C2124</td>
<td>0020 00 0 02131</td>
<td>TRA</td>
<td>AR2</td>
<td>GO TO EXIT.</td>
<td>TT154870</td>
</tr>
<tr>
<td>O2125</td>
<td>0500 00 1 02166</td>
<td>OPFND CAL</td>
<td>TABL,1</td>
<td>OPERATION FOUND, PICKUP WORD.</td>
<td>TT154880</td>
</tr>
<tr>
<td>O2126</td>
<td>0320 00 0 03140</td>
<td>ANA</td>
<td>*G00000077777</td>
<td>TT154890</td>
<td></td>
</tr>
<tr>
<td>O2127</td>
<td>0767 00 0 00006</td>
<td>ALS</td>
<td>6</td>
<td>TT154900</td>
<td></td>
</tr>
<tr>
<td>C2130</td>
<td>0501 00 1 03156</td>
<td>ORA</td>
<td>*H,000</td>
<td>TT154910</td>
<td></td>
</tr>
<tr>
<td>C2131</td>
<td>0774 00 2 00000</td>
<td>AR2</td>
<td>AXT</td>
<td>*;2</td>
<td>TT154920</td>
</tr>
<tr>
<td>C2132</td>
<td>0774 00 1 00000</td>
<td>AR1</td>
<td>AXT</td>
<td>*;1</td>
<td>TT154930</td>
</tr>
<tr>
<td>C2133</td>
<td>0020 00 4 00001</td>
<td>TRA</td>
<td>1,4</td>
<td>RETURN TO CALLER.</td>
<td>TT154940</td>
</tr>
<tr>
<td>C2134</td>
<td>0 00000 0 00000</td>
<td>PRE</td>
<td>PZE</td>
<td>TT154950</td>
<td></td>
</tr>
<tr>
<td>C2135</td>
<td>0 00000 0 00000</td>
<td>OPBIN</td>
<td>PZE</td>
<td>TT154970</td>
<td></td>
</tr>
</tbody>
</table>

#### SPACE 2

**TABLES FOR (OPCD).**

<table>
<thead>
<tr>
<th>LSTA BCI</th>
<th>TXL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCI 1</td>
<td>TXL</td>
</tr>
<tr>
<td>BCI 1</td>
<td>STR</td>
</tr>
<tr>
<td>BCI 1</td>
<td>MZE</td>
</tr>
<tr>
<td>BCI 1</td>
<td>TXM</td>
</tr>
<tr>
<td>BCI 1</td>
<td>TXI</td>
</tr>
<tr>
<td>BCI 1</td>
<td>PZE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABL VFD</th>
<th>012/0000,H24/DHTR</th>
<th>HTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFD 012/0020,H24/OTRA</td>
<td>TRA</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0021,H24/OTTR</td>
<td>TRA</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0022,H24/OTRC</td>
<td>TRA</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0024,H24/OTRC</td>
<td>TRA</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0030,H24/OTEF</td>
<td>TEFA</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0031,H24/OTEF</td>
<td>TEFC</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0040,H24/OITL</td>
<td>TLQ</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0041,H24/OITL</td>
<td>IIA</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0042,H24/OIIO</td>
<td>T IO</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0043,H24/OAI</td>
<td>OAI</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0044,H24/OAI</td>
<td>OAI</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0045,H24/OAI</td>
<td>OAI</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0046,H24/OAI</td>
<td>OAI</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0051,H24/011R</td>
<td>IIR</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0053,H24/01IR</td>
<td>IIR</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0054,H24/01IR</td>
<td>IIR</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0055,H24/01IR</td>
<td>IIR</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0056,H24/01IR</td>
<td>IIR</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0057,H24/01IR</td>
<td>IIR</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0060,H24/OTCO</td>
<td>TCGA</td>
<td></td>
</tr>
<tr>
<td>VFD 012/0061,H24/OTCO</td>
<td>TCOB</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>02172</td>
<td>006200632346</td>
<td>VFD</td>
</tr>
<tr>
<td>02173</td>
<td>006300632346</td>
<td>VFD</td>
</tr>
<tr>
<td>02174</td>
<td>007400636267</td>
<td>VFD</td>
</tr>
<tr>
<td>02175</td>
<td>010000637125</td>
<td>VFD</td>
</tr>
<tr>
<td>02176</td>
<td>011400236551</td>
<td>VFD</td>
</tr>
<tr>
<td>02177</td>
<td>012000634743</td>
<td>VFD</td>
</tr>
<tr>
<td>02200</td>
<td>0130100672321</td>
<td>VFD</td>
</tr>
<tr>
<td>02201</td>
<td>014000634665</td>
<td>VFD</td>
</tr>
<tr>
<td>02202</td>
<td>016100635046</td>
<td>VFD</td>
</tr>
<tr>
<td>02203</td>
<td>016200635047</td>
<td>VFD</td>
</tr>
<tr>
<td>02204</td>
<td>020000044770</td>
<td>VFD</td>
</tr>
<tr>
<td>02205</td>
<td>020400654344</td>
<td>VFD</td>
</tr>
<tr>
<td>02206</td>
<td>022000266530</td>
<td>VFD</td>
</tr>
<tr>
<td>02207</td>
<td>022100266547</td>
<td>VFD</td>
</tr>
<tr>
<td>02210</td>
<td>022400652430</td>
<td>VFD</td>
</tr>
<tr>
<td>02211</td>
<td>022500652447</td>
<td>VFD</td>
</tr>
<tr>
<td>02212</td>
<td>024000262431</td>
<td>VFD</td>
</tr>
<tr>
<td>02213</td>
<td>024100262431</td>
<td>VFD</td>
</tr>
<tr>
<td>02214</td>
<td>026000264447</td>
<td>VFD</td>
</tr>
<tr>
<td>02215</td>
<td>030000262124</td>
<td>VFD</td>
</tr>
<tr>
<td>02216</td>
<td>030200266222</td>
<td>VFD</td>
</tr>
<tr>
<td>02217</td>
<td>030400262444</td>
<td>VFD</td>
</tr>
<tr>
<td>02220</td>
<td>030600262424</td>
<td>VFD</td>
</tr>
<tr>
<td>02221</td>
<td>032000214562</td>
<td>VFD</td>
</tr>
<tr>
<td>02222</td>
<td>032200255121</td>
<td>VFD</td>
</tr>
<tr>
<td>02223</td>
<td>034000231262</td>
<td>VFD</td>
</tr>
<tr>
<td>02224</td>
<td>036100213243</td>
<td>VFD</td>
</tr>
<tr>
<td>02225</td>
<td>040000212424</td>
<td>VFD</td>
</tr>
<tr>
<td>02226</td>
<td>040100212444</td>
<td>VFD</td>
</tr>
<tr>
<td>02227</td>
<td>040200626622</td>
<td>VFD</td>
</tr>
<tr>
<td>02230</td>
<td>040200307451</td>
<td>VFD</td>
</tr>
<tr>
<td>02231</td>
<td>044000313162</td>
<td>VFD</td>
</tr>
<tr>
<td>02232</td>
<td>044100432431</td>
<td>VFD</td>
</tr>
<tr>
<td>02233</td>
<td>044200462311</td>
<td>VFD</td>
</tr>
<tr>
<td>02234</td>
<td>044400462663</td>
<td>VFD</td>
</tr>
<tr>
<td>02235</td>
<td>044500513162</td>
<td>VFD</td>
</tr>
<tr>
<td>02236</td>
<td>044600464563</td>
<td>VFD</td>
</tr>
<tr>
<td>02237</td>
<td>046000432421</td>
<td>VFD</td>
</tr>
<tr>
<td>02240</td>
<td>050500234321</td>
<td>VFD</td>
</tr>
<tr>
<td>02241</td>
<td>050200234326</td>
<td>VFD</td>
</tr>
<tr>
<td>02242</td>
<td>052000712563</td>
<td>VFD</td>
</tr>
<tr>
<td>02243</td>
<td>052200672523</td>
<td>VFD</td>
</tr>
<tr>
<td>02244</td>
<td>053400436721</td>
<td>VFD</td>
</tr>
<tr>
<td>02245</td>
<td>053500432123</td>
<td>VFD</td>
</tr>
<tr>
<td>02246</td>
<td>054000512330</td>
<td>VFD</td>
</tr>
<tr>
<td>02247</td>
<td>054100512330</td>
<td>VFD</td>
</tr>
<tr>
<td>02250</td>
<td>054400323230</td>
<td>VFD</td>
</tr>
<tr>
<td>02251</td>
<td>054500432330</td>
<td>VFD</td>
</tr>
<tr>
<td>02252</td>
<td>056000432450</td>
<td>VFD</td>
</tr>
<tr>
<td>02253</td>
<td>056200431511</td>
<td>VFD</td>
</tr>
<tr>
<td>02254</td>
<td>056400254922</td>
<td>VFD</td>
</tr>
<tr>
<td>02255</td>
<td>060000262436</td>
<td>VFD</td>
</tr>
<tr>
<td>02256</td>
<td>060200262346</td>
<td>VFD</td>
</tr>
<tr>
<td>02257</td>
<td>060400262331</td>
<td>VFD</td>
</tr>
<tr>
<td>O2261</td>
<td>062100626321</td>
<td>VFD</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-----</td>
</tr>
<tr>
<td>O2262</td>
<td>062200626324</td>
<td>VFD</td>
</tr>
<tr>
<td>O2263</td>
<td>062500626363</td>
<td>VFD</td>
</tr>
<tr>
<td>O2264</td>
<td>063000626347</td>
<td>VFD</td>
</tr>
<tr>
<td>O2265</td>
<td>063400626721</td>
<td>VFD</td>
</tr>
<tr>
<td>O2266</td>
<td>064000626330</td>
<td>VFD</td>
</tr>
<tr>
<td>O2267</td>
<td>064100626330</td>
<td>VFD</td>
</tr>
<tr>
<td>O2270</td>
<td>070000623727</td>
<td>VFD</td>
</tr>
<tr>
<td>O2271</td>
<td>073400624717</td>
<td>VFD</td>
</tr>
<tr>
<td>O2272</td>
<td>073700742123</td>
<td>VFD</td>
</tr>
<tr>
<td>O2273</td>
<td>075400476721</td>
<td>VFD</td>
</tr>
<tr>
<td>O2274</td>
<td>076000476225</td>
<td>VFD</td>
</tr>
<tr>
<td>O2275</td>
<td>076100456487</td>
<td>VFD</td>
</tr>
<tr>
<td>O2276</td>
<td>076200512462</td>
<td>VFD</td>
</tr>
<tr>
<td>O2277</td>
<td>076300434362</td>
<td>VFD</td>
</tr>
<tr>
<td>O2300</td>
<td>076400226251</td>
<td>VFD</td>
</tr>
<tr>
<td>O2301</td>
<td>076500435162</td>
<td>VFD</td>
</tr>
<tr>
<td>O2302</td>
<td>076600665162</td>
<td>VFD</td>
</tr>
<tr>
<td>O2303</td>
<td>076700214362</td>
<td>VFD</td>
</tr>
<tr>
<td>O2304</td>
<td>077000662526</td>
<td>VFD</td>
</tr>
<tr>
<td>O2305</td>
<td>077100215162</td>
<td>VFD</td>
</tr>
<tr>
<td>O2306</td>
<td>077200512566</td>
<td>VFD</td>
</tr>
<tr>
<td>O2307</td>
<td>077400216763</td>
<td>VFD</td>
</tr>
<tr>
<td>O2310</td>
<td>077600022445</td>
<td>VFD</td>
</tr>
<tr>
<td>O2311</td>
<td>402100254563</td>
<td>VFD</td>
</tr>
<tr>
<td>O2312</td>
<td>402200635123</td>
<td>VFD</td>
</tr>
<tr>
<td>O2313</td>
<td>402400635123</td>
<td>VFD</td>
</tr>
<tr>
<td>O2314</td>
<td>403000632526</td>
<td>VFD</td>
</tr>
<tr>
<td>O2315</td>
<td>403100632526</td>
<td>VFD</td>
</tr>
<tr>
<td>O2316</td>
<td>404000631512</td>
<td>VFD</td>
</tr>
<tr>
<td>O2317</td>
<td>404600473121</td>
<td>VFD</td>
</tr>
<tr>
<td>O2320</td>
<td>405100313143</td>
<td>VFD</td>
</tr>
<tr>
<td>O2321</td>
<td>405400432663</td>
<td>VFD</td>
</tr>
<tr>
<td>O2322</td>
<td>405500632143</td>
<td>VFD</td>
</tr>
<tr>
<td>O2323</td>
<td>405600434563</td>
<td>VFD</td>
</tr>
<tr>
<td>O2324</td>
<td>405700513143</td>
<td>VFD</td>
</tr>
<tr>
<td>O2325</td>
<td>406000632345</td>
<td>VFD</td>
</tr>
<tr>
<td>O2326</td>
<td>406100632345</td>
<td>VFD</td>
</tr>
<tr>
<td>O2327</td>
<td>406200632345</td>
<td>VFD</td>
</tr>
<tr>
<td>O2330</td>
<td>406300632345</td>
<td>VFD</td>
</tr>
<tr>
<td>O2331</td>
<td>410000634571</td>
<td>VFD</td>
</tr>
<tr>
<td>O2332</td>
<td>411400232150</td>
<td>VFD</td>
</tr>
<tr>
<td>O2333</td>
<td>412000634341</td>
<td>VFD</td>
</tr>
<tr>
<td>O2334</td>
<td>413000672343</td>
<td>VFD</td>
</tr>
<tr>
<td>O2335</td>
<td>414000634566</td>
<td>VFD</td>
</tr>
<tr>
<td>O2336</td>
<td>415400239150</td>
<td>VFD</td>
</tr>
<tr>
<td>O2337</td>
<td>420000444751</td>
<td>VFD</td>
</tr>
<tr>
<td>O2340</td>
<td>426000642644</td>
<td>VFD</td>
</tr>
<tr>
<td>O2341</td>
<td>430000642621</td>
<td>VFD</td>
</tr>
<tr>
<td>O2342</td>
<td>430200642662</td>
<td>VFD</td>
</tr>
<tr>
<td>O2343</td>
<td>430400642144</td>
<td>VFD</td>
</tr>
<tr>
<td>O2344</td>
<td>430600642444</td>
<td>VFD</td>
</tr>
<tr>
<td>O2345</td>
<td>432000214521</td>
<td>VFD</td>
</tr>
<tr>
<td>O2346</td>
<td>434000432162</td>
<td>VFD</td>
</tr>
<tr>
<td>O2347</td>
<td>440000622244</td>
<td>VFD</td>
</tr>
</tbody>
</table>
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 5, SVC0N, PMR, OCT, OCTADR, AND TABBK.

02350  4500000323143  VFD  012/4500,H24/OCAL  CAL  TT56400
02351  45010004651121  VFD  012/4501,H24/ODRA  ORA  TT56410
02352  4520000457163  VFD  012/4520,H24/ONZT  NZT  TT56420
02353  4534000434726  VFD  012/4534,H24/OLXD  LXD  TT56430
02354  4535000432423  VFD  012/4535,H24/OLDC  LDC  TT56440
02355  4540000512330  VFD  012/4540,H24/ORCH  RCHB  TT56450
02356  4541000912330  VFD  012/4541,H24/ORCH  RCHB  TT56460
02357  4544000432330  VFD  012/4544,H24/OLCH  LCHB  TT56470
02360  4545000432330  VFD  012/4545,H24/OLCH  LCHB  TT56480
02361  4564000437471  VFD  012/4564,H24/OLFI  LPI  TT56490
02362  4600000626350  VFD  012/4600,H24/OSIQ  STQ  TT56500
02363  4601000625131  VFD  012/4601,H24/OSRI  SRI  TT56510
02364  4602000465162  VFD  012/4602,H24/ORSI  ORS  TT56520
02365  4604000624731  VFD  012/4604,H24/OSPI  SPI  TT56530
02366  4620000624350  VFD  012/4620,H24/OSLQ  SLQ  TT56540
02367  4625000623430  VFD  012/4625,H24/OSTL  STL  TT56550
02368  4634000626724  VFD  012/4634,H24/OSLD  SD  TT56560
02369  4640000622330  VFD  012/4640,H24/OSCH  SCHB  TT56570
02370  4641000622330  VFD  012/4641,H24/OSCH  SCHB  TT56580
02371  4700000232143  VFD  012/4700,H24/OCAD  CAD  TT56590
02372  4701000023124  VFD  012/4713,H24/OPDX  PDX  TT56600
02373  4713000472647  VFD  012/4737,H24/OPDC  PDC  TT56610
02374  4734000046265  VFD  012/4756,H24/ONSE  OSE  TT56620
02375  4754000476246  VFD  012/4763,H24/OLGL  LG  TT56630
02376  4763000432743  VFD  012/4763,H24/OLGL  LG  TT56640
02400  4763000432743  VFD  012/4763,H24/OLGL  LG  TT56650
02401  4764000226226  VFD  012/4764,H24/OBDF  BSF  TT56660
02402  4765000437591  VFD  012/4765,H24/OLGR  LG  TT56670
02403  4772000516455  VFD  012/4772,H24/OLUS  RUS  TT56680
02404  477300051043  VFD  012/4773,H24/OLUS  RUS  TT56690
02405  4774000216723  VFD  012/4774,H24/OCAX  AX  TT56700
02406  775400000222  VFD  18,1-NUM  TT56710
02406  0 77540 0 00022  SIZE PZE  TT56720

NB ADDRESS OF SIZE IS (2**E+2), WHERE E IS A NUMBER SUCH THAT TT56730
(2**E) .GT. (NUMBER OF ENTRIES IN TABLE) .GT. (2**E-1)) TT56740
DECREMENT OF SIZE IS COMPLEMENT OF TABLE LENGTH. TT56750

00240  NUM EQU ENTBL-TA B NUMBE R IN OP TABLE. TT56760

33
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.

SECTION 6, INPUT/OUTPUT SUBROUTINES.

TTL  SECTION 6, INPUT/OUTPUT SUBROUTINES  TTL  TTI6000
    BUFFERED NOT, CALLING SEQUENCE,  TTI60010
    TSX $WOT, 4 OR TSX $WOT, 4  TTI60020
    PZE FIRST, T, LAST  TTI60030
    PZE FIRST, T, N  TTI60040
IF T IS NON-ZERO OR IF SWS IS DOWN, PRINT ALSO.
    WRITES ONLY FIRST 22 WORDS ON TAPE.  TTI60050
    TTI60060
    TTI60070
    TTI60080
    TTI60090
    TTI60100
    TTI61000
    TTI61010
    TTI61100

02407  06 34 00 0 02442  WOT SXL WTX4, 4  SAVE IRR.
02410  00 60 00 0 02410  TCUA *  WAIT FOR DSC A.
02411  07 60 00 0 01000  ETUA CHECK FOR ECT CHN A.
02412  00 20 00 0 02452  TRA ETA  EOT, EXIT.
02413  05 00 00 0 00000  CAL 1, 4  GET CONTROL WORD.
02414  06 25 00 0 02451  STT WTXA  SAVE TAG FOR DECISION TO PRINT.
02415  07 34 00 0 04000  PAX 0, 4  FIRST TC IRR.
02416  06 34 00 0 02432  SXX WTX1, 4  SAVE FOR BUFFERING.
02417  06 34 00 0 02422  SXX WTX1, 4  SAVE FOR FINDING N.
02420  07 34 00 0 04000  PDX 0, 4  N OR LAST TC IRR.
02421  00 00 00 0 02422  TXI **1, 4, 1  PLUS 1.
02422  00 0000 0 02424  WTXI TXI **2, 4, **  IS IT N OR LAST.
02423  1 77 77 4 02424  TXI **1, 4, 1  N IN IRR.
02424  3 00 26 0 02426  TXX **2, 4, 22  N  LE, 22.
02425  07 74 00 0 00026  AXT 22, 4  N = 22.
02426  06 34 00 0 02450  SXX WTCOM, 4  SAVE FOR DSC COMMAND.
02427  02 71 00 0 02430  TXI **1, 4, 1  FORM BUFFER.
02430  06 34 00 0 02436  SXX WTXSL, 4  SAVE FOR BUFFERING.
02431  06 34 00 0 02450  LXX WTCOM, 4  N IN IRR.
02432  00 00 00 0 02433  WTXI TXI **1, 4, **  FORM FIRST N.
02433  06 34 00 0 02435  SXL WTCAL, 4  SAVE FOR BUFFERING.
02434  05 34 00 0 02450  LXX WTCOM, 4  N IN IRR.
02435  05 00 00 0 04000  WICAL CAL **4  MOVE TO BUFFER.
02436  06 02 00 0 04000  WTXSL SLW **4  **
02437  00 00 01 0 02435  TIX -2, 4, 1  **
02440  07 66 00 0 01203  WTXA WTX 3  SELECT OUTPUT TAPE.
02441  05 40 00 0 02450  RCHA WTCOM  LOAD TO WRITE OUT BUFFER.
02442  07 74 00 0 04000  WTX4 AXT **4  RESTORE IRR.
02443  05 20 00 0 02451  ZET WTXA  WAS THE TAG ZERO.
02444  02 00 00 0 02520  TRA PRINT  NO, THEN PRINT ALSO.
02445  07 60 00 0 00165  SWT 5  ZERO TAG, BUT CHECK SWS.
02446  02 00 00 0 04000  TRA 2, 4  UP, DON'T PRINT.
02447  02 00 00 0 02520  TRA PRINT  DOWN, PRINT ALSO.
02450  00 00 00 0 02715  WTXCOM IOC  BUFF, 0, **  OUTPUT COMMAND, N IN DECREMENT.
02451  00 00 00 0 00000  WTXTAG PZE 0, 0  STORAGE FOR TAG.

02452  07 64 00 0 01202  ETA BSFA 2  END OF TAPE A3, SET FOR RESTART.
02453  07 64 00 0 02204  BSFB 4  REMOVE BINARY OUTPUT.
02454  07 64 00 0 02204  BSFB 4  **
02455  07 74 0 1 00012  AXT 10, 1  10 CLOSE COMMENTS.
02456  07 74 0 0 04007  TSX WOT, 4  CLOSE COMMENT.
02457  0 00003 0 01077  PZE ENDT, 0 3  **
02460  0 00003 0 04007  TSX WOT, 4  CHANNEL 4 SKIP.
02461  0 00001 0 03147  PZE H4, 0, 1  **
02462  0 00003 0 01203  WEFA 3  END OF FILE MARK.
TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 6. INPUT/OUTPUT SUBROUTINES.

02463  2 00001 1 02456
02464 -0772 00 0 01203
02465  0411 00 0 03125
02466  0760 00 0 01000
02467  0761 00 0 00000
02470  0074 00 4 02520
02471  000022 0 02476
02472  0760 00 0 01000
02473  0020 00 0 00765
02474 -0764 00 0 01203
02475  0020 00 0 02472
02476  002545246046
02477  266063214725
02500  604421514260
02501  254523466445
02502  632551252460
02503  464560632147
02504  256021037360
02505  632147256023
02506  434662254233
02507  602330214527
02510  256036214725
02511  602145246047
02512  512562626062
02513  632151636063
02514  466051256263
02515  215163606330
02516  316260444622
02517  336060606060

SPACE

TSX $PRINT,4 OR TSX $PRINT,4
PRE FIRST,T,LAST PRE FIRST,T,N
RECOGNIZES BLANK, 0, AND 1 AS CARRIAGE CONTROL (OTHER CARRIAGE CONTROLS DON'T WORK TOO WELL), OTHERWISE, SINGLE SPACE. PRINTS ONLY 20 WORDS, AND T IS IGNORED.
PRE=ZE, NORMAL RETURN (2,4), PRE=MZE, SPACE 0.1 PAGE, LIGHT UP CONSOLE, AND STOP (HP= -1,7,63), RESTART RETURNS (2,4).

02520  0634 00 4 02642
02521  0634 00 2 02643
02522  0634 00 1 02644
02523  0600 00 0 00005
02524  -0061 00 0 02526
02525  -0540 00 0 00261
02526 -0500 00 4 00001
02527  0734 00 4 00000
02530  -0634 00 4 02541

PRINT SKA

PR4,4 SAVE IRS.
SXA PR2,2
SXA PR1,1
STZ 5 RESET INTERVAL TIMER.
TCTN *+2 IF TAPE TIMER IN USE.
RCHT QUIT RESET TAPE TIMER.
CAL 1,4 GET CONTROL WORD.
PAX 0,4 FIRST TO IR4.
SAX PTX1,4 SAVE FOR PICKUP.

WEOTA3 8CI 9,0END OF TAPE MARK ENCOUNTERED ON TAPE A3, TAPE CLOSED. TT160680

BCI 9, CHANGE TAPE AND PRESS START TO RESTART THIS JOB. TT160690

TT160550
TT160560
TT160570
TT160580
TT160590
TT160600
TT160610
TT160620
TT160630
TT160640
TT160650
TT160660
TT160670
TT160680
TT160690
TT160700
TT160710
TT160720
TT160730
TT160740
TT160750
TT160760
TT160770
TT160780
TT160790
TT160800
TT160810
TT160820
TT160830
TT160840
TT160850
TT160860
TT160870
TT160880
TT160890
TT160900
TT160910
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 6, INPUT/OUTPUT SUBROUTINES.

02531 -0634 00 4 02534  SXD *+3,4 SAVE FOR FINDING N.  TT160920
02532 -0734 00 4 00000  PDX 0,4  N OR LAST TO IR4.  TT160930
02533 1 00001 4 02534  TXI *+1,4,1 PLUS 1.  TT160940
02534 2 00000 4 02536  TXI *+2,4,** IS IT N OR LAST.  TT160950
02535 1 7777 4 02536  TXI *+1,4,1 N IN IR4.  TT160960
02536 -3 00024 4 02540  TXL *+2,4,20 IS N .LE. 20.  TT160970
02537 0774 00 00024  AXT 20,4  N = 20.  TT160980
02540 0634 00 4 02614  SXR WDCNT,4 SAVE WDCNT.  TT160990
02541 1 00000 4 02542  PTXI TXI *+1,4,** COMPUTE LAST+1.  TT161000
02542 0634 00 4 02616  SXR LDQ,4 INSERT IN PICKUP.  TT161010
02543 0534 00 4 02614  LXR WDCNT,4 N IN IR4.  TT161020
02544 0522 00 0 02616  XEC LDQ PICKUP FIRST WORD.  TT161030
02545 -0734 00 00000  PXD 0,0 GET FIRST CHARACTER.  TT161040
02546 -0763 00 0 00006  LGL 6 FOR CARRIAGE CONTROL.  TT161050
02547 -0340 00 0 03126  LAS +1 COMPARISON WITH 1.  TT161060
02550 -0500 00 0 03127  CAL +2 C.C. ,G, SINGLE SPACE.  TT161070
02551 -0020 00 0 02552  PAC 0,3 C.C. ,E, NEW PAGE.  TT161080
02552 0737 00 0 00000  PAC 0,2 C.C. ,E, , DOUBLE SPACE.  TT161090
02553 -0500 00 0 02660  CAL SPRS,2 PICKUP SPR.  TT161100
02554 0602 00 0 02633 SLW SPR AND INSERT AFTER SELECT.  TT161110
02555 0774 00 4 00005  AXT 5,4 SET FOR FIVE CHARACTERS, FIRST WORD.  TT161120
02556 -0500 00 0 03511 CAL 020000000000 COLUMN MARK, SKIP 1ST COLUMN.  TT161130
02557 0774 00 1 00030 HRITA AXT 24,1 CLEAR CARD IMAGE.  TT161140
02558 0600 00 1 02715 STZ CARDIM+24,1 **.  TT161150
02559 0600 00 1 02716 STZ CARDIM+25,1 **.  TT161160
02562 2 00002 1 02560 1 02650 TIX *+2,1,2 **.  TT161170
02563 0774 00 2 00001 AXT 1,2 SET MARK FOR LEFT HALF.  TT161180

02564 0602 00 0 02657 PRLP SLW PRCOL SAVE COLUMN MARKER.  TT161190
02565 -0754 00 0 00000 PXU 0,0 GET NEXT CHARACTER.  TT161200
02566 -0763 00 0 00006 LGL 6 **.  TT161210
02567 0767 00 0 00001 ALS 1 DOUBLE IT.  TT161220
02570 0734 00 1 00000 PAX 0,1 AND PLACE IN IRL.  TT161230
02571 -0500 00 0 02657 CAL PRCOL GET COLUMN MARKER.  TT161240

02572 -3 00037 1 02606 ZONE TXL DIG,1,31 IF NO ZONE, SKIP.  TT161250
02573 -3 00137 1 02575 TXL +2,1,95 IGNORE BLANK.  TT161260
02574 -3 00140 1 02612 TXL RLOOP,1,96 **.  TT161270
02575 3 00077 1 02600 TXH +3,1,63 CHECK FOR 12 ZONE.  TT161280
02576 -0602 00 2 02714 ORS CARDIM+23,2 YES, OR IT IN.  TT161290
02577 1 77740 1 02605 TXI TZEDG,1,32 REMOVE ZONE, AND CHECK DIGIT.  TT161300
02560 3 00137 1 02603 TXH +3,1,95 CHECK FOR 11 ZONE.  TT161310
02601 -0602 00 2 02712 ORS CARDIM+21,2 YES, OR IT IN.  TT161320
02602 1 77700 1 02605 TXI TZEDG,1,64 REMOVE ZONE AND CHECK DIGIT.  TT161330
02603 -0602 00 2 02710 ORS CARDIM+19,2 OR IN O ZONE.  TT161340
02604 1 77640 1 02605 TXI +1,1,-96 REMOVE ZONE, AND CHECK DIGIT.  TT161350
02605 -3 00000 1 02612 TZE TG TLOCP,1,0 IGNORE O DIGIT WITH A ZONE.  TT161360
02606 -3 00022 1 02611 CIG TXL *+3,1,18 CHECK FOR (8-N) CHARACTER.  TT161370
02607 -0602 00 2 02670 ORS CARDIM+3,2 YES, OR IN B.  TT161380
02610 1 77760 1 02611 TXI *+1,1,-16 REMOVE 8.  TT161390
### TESTS FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM

#### SECTION 6. INPUT/OUTPUT SUBROUTINES

```assembly
test 1 02611 -0602 00 3 02710 02612 0771 00 0 00001 02613 2 00001 4 02564 02614 0774 00 4 00000 02615 -2 00001 4 02625 02616 0 0560 0 4 00000 02617 0634 0 4 02614 02618 0774 0 4 00006 02619 -0100 0 0 02564 02620 -3 00000 2 02625 02621 -0500 0 0 03153 02622 1 7777 2 02564 02623 PNOW AXT 24,1 02624 0774 0 1 00030 02625 0060 0 0 02626 02627 -0500 0 0 02715 02628 0602 0 1 02745 02629 2 00001 1 02630 02631 02632 0766 0 0 01361 02633 0000 60 0 02633 02634 0540 0 0 02643 02635 -3 00001 4 02642 02636 -0500 0 0 02663 02637 0602 0 0 02633 02640 -0500 0 0 03153 02641 0020 0 0 02557 02642 0774 0 4 00000 02643 0774 0 4 00000 02644 0774 0 1 00000 02645 5000 0 4 00001 02646 0120 0 4 00002 02647 0766 0 0 01361 02648 0760 0 0 01364 02649 0760 0 0 01363 02650 0560 0 0 03125 02651 0760 0 0 00000 02652 0560 0 0 03166 02653 0760 0 0 01371 02654 00003 0 02715 02655 0420 0 7 7777 02656 0020 0 4 00002 02657 0 00000 0 00000 02658 0 00000 0 00000 02659 0 00000 0 00000 02660 0 00000 0 00000 02661 0 00000 0 00000 02662 0 00000 0 00000 02663 0 00000 0 00000 02664 0 00000 0 00000
```

#### STORAGE AND CONSTANTS FOR PRINT

```assembly
02520 NPRINT SYN PRINT EXTERNAL NAME FOR PRINT.
```

---

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>02611</td>
<td>-0602</td>
<td>00 3 02710</td>
</tr>
<tr>
<td>02612</td>
<td>0771</td>
<td>00 0 00001</td>
</tr>
<tr>
<td>02613</td>
<td>2 00001</td>
<td>4 02564</td>
</tr>
<tr>
<td>02614</td>
<td>0774</td>
<td>00 4 00000</td>
</tr>
<tr>
<td>02615</td>
<td>-2 00001</td>
<td>4 02625</td>
</tr>
<tr>
<td>02616</td>
<td>0 0560</td>
<td>0 4 00000</td>
</tr>
<tr>
<td>02617</td>
<td>0634</td>
<td>0 4 02614</td>
</tr>
<tr>
<td>02618</td>
<td>0774</td>
<td>0 4 00006</td>
</tr>
<tr>
<td>02619</td>
<td>-0100</td>
<td>0 0 02564</td>
</tr>
<tr>
<td>02620</td>
<td>-3 00000</td>
<td>2 02625</td>
</tr>
<tr>
<td>02621</td>
<td>-0500</td>
<td>0 0 03153</td>
</tr>
<tr>
<td>02622</td>
<td>1 7777</td>
<td>2 02564</td>
</tr>
<tr>
<td>02623</td>
<td>PNOW AXT</td>
<td>24,1</td>
</tr>
<tr>
<td>02624</td>
<td>0774</td>
<td>0 1 00030</td>
</tr>
<tr>
<td>02625</td>
<td>0060</td>
<td>0 0 02626</td>
</tr>
<tr>
<td>02626</td>
<td>-0500</td>
<td>0 0 02715</td>
</tr>
<tr>
<td>02627</td>
<td>0602</td>
<td>0 1 02745</td>
</tr>
<tr>
<td>02628</td>
<td>2 00001</td>
<td>1 02627</td>
</tr>
<tr>
<td>02629</td>
<td>0766</td>
<td>0 0 01361</td>
</tr>
<tr>
<td>02630</td>
<td>0000</td>
<td>60 0 02633</td>
</tr>
<tr>
<td>02631</td>
<td>0540</td>
<td>0 0 02643</td>
</tr>
<tr>
<td>02632</td>
<td>-3 00001</td>
<td>4 02642</td>
</tr>
<tr>
<td>02633</td>
<td>-0500</td>
<td>0 0 02663</td>
</tr>
<tr>
<td>02634</td>
<td>0602</td>
<td>0 0 02633</td>
</tr>
<tr>
<td>02635</td>
<td>-0500</td>
<td>0 0 03153</td>
</tr>
<tr>
<td>02636</td>
<td>0020</td>
<td>0 0 02557</td>
</tr>
<tr>
<td>02637</td>
<td>0774</td>
<td>0 4 00000</td>
</tr>
<tr>
<td>02638</td>
<td>0774</td>
<td>0 2 00000</td>
</tr>
<tr>
<td>02639</td>
<td>0774</td>
<td>0 1 00000</td>
</tr>
<tr>
<td>02640</td>
<td>0500</td>
<td>0 4 00001</td>
</tr>
<tr>
<td>02641</td>
<td>0120</td>
<td>0 4 00002</td>
</tr>
<tr>
<td>02642</td>
<td>0766</td>
<td>0 0 01361</td>
</tr>
<tr>
<td>02643</td>
<td>0760</td>
<td>0 0 01364</td>
</tr>
<tr>
<td>02644</td>
<td>0760</td>
<td>0 0 01363</td>
</tr>
<tr>
<td>02645</td>
<td>0560</td>
<td>0 0 03125</td>
</tr>
<tr>
<td>02646</td>
<td>0760</td>
<td>0 0 00000</td>
</tr>
<tr>
<td>02647</td>
<td>0560</td>
<td>0 0 03166</td>
</tr>
<tr>
<td>02648</td>
<td>0760</td>
<td>0 0 01371</td>
</tr>
<tr>
<td>02649</td>
<td>00003</td>
<td>0 02715</td>
</tr>
<tr>
<td>02650</td>
<td>0420</td>
<td>0 7 7777</td>
</tr>
<tr>
<td>02651</td>
<td>0020</td>
<td>0 4 00002</td>
</tr>
<tr>
<td>02652</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02653</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02654</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02655</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02656</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02657</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02658</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02659</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02660</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02661</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02662</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02663</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02664</td>
<td>0 00000</td>
<td>0 00000</td>
</tr>
<tr>
<td>02520</td>
<td>NPRINT SYN PRINT EXTERNAL NAME FOR PRINT.</td>
<td></td>
</tr>
</tbody>
</table>
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
SECTION 6, INPUT/OUTPUT SUBROUTINES.

SPACE  2
STORAGE AREA.

02665  CARCIM BSS  24  CARD IMAGE BUFFER FOR PRINT.
02715  BUFF BSS   24  OUTPUT BUFFER FOR WCT.
02745  OUT BSS    22  OUTPUT BUFFER FOR POST MORTEM.

00036  NSYM EQU  30  30 NAMES ALLOWED IN MOVIE).
02773  NAME BSS   NSYM SUBROUTINE NAME TABLE.
03031  ORIGIN BSS NSYM SUBROUTINE ORIGIN TABLE.
03067  ENTRY BSS  NSYM SUBROUTINE ENTRY POINT TABLE.

END

LITERALS
03125  000000000000
03126  000000000001
03127  000000000002
03130  000000000003
03131  000000000005
03132  000000000006
03133  000000000012
03134  000000000072
03135  000000000073
03136  000000000454
03137  00000602646
03140  0000777777
03141  0001700000
03142  004623632143
03143  006060606060
03144  007070707000
03145  014623632143
03146  016060606060
03147  046060606060
03150  070707070777
03151  200000000000
03152  377700000000
03153  400000000000
03154  462660000000
03155  600073073000
03156  606000000060
03157  606060604000
03160  606060606060
03161  632562636260
03162  700000000000
03163  744421314534
03164  777400000000
03165  777700000000
03166  777777777777
REFERENCES TO DEFINED SYMBOLS

2223  T  136, 175, 204, 205, 704, 705, 710, 1147, 1150, 2524, 2525
2223  AC  1157, 1535
2223  MQ  1156, 1541
1653  ADR  1737
2132  AR1  2050
2131  AR2  2047, 2124
2131  CAP  215
2606  DIG  2572
2020  DX1  1642
2017  DX2  1641
2016  DX4  1640
2452  ETA  2412
1776  FIN  1746
2557  HRI  2641
2023  ILC  1645, 1653, 1734, 1736, 1772, 1774
1164  IRI  1154, 1551
1165  IR2  1155, 1554
1166  IR4  220, 263, 315, 426, 432, 525, 534, 630, 1145, 1557
2616  LDO  2542, 2544
32  MAP
240  NUM  2062, 2406, 2467
1260  GCT  1301, 1536, 1542, 1546, 1715, 1743
234  ORG  214, 1464
1503  PMH  1424, 1427, 1441, 1443, 1461
1401  PMR  712, 713
1750  PPM
2644  PRI  2522
2643  PR2  2521
2642  PR4  2520, 2635
2134  PRE  2055, 2056
1740  REG  1652, 1661, 1674
1  RIP  201
533  SRL  173
473  SSI  406, 416, 433
1145  SX4
235  TIT  202, 707
425  TRR  405
250  WEP  226, 230, 232
2407  WGT  67, 71, 121, 134, 135, 175, 231, 276, 344, 361, 440, 445, 451, 467, 565, 600
645, 714, 745, 747, 753, 755, 1401, 1442, 1462, 1470, 1562, 1564, 1566, 1570, 1572, 1574, 1576
1600, 1730, 1732, 1760, 2014, 2456, 2460
702  BACK  233, 301, 347, 364, 472, 570, 603, 650
2715  BUFF  2427, 2450, 2630, 2664
1445  CD1  1414, 1417
167  C(1)  27
170  C(2)  31
171  C(7)  33
172  C(8)  35
1523  DCGN  1411
765  DGR  722, 743, 1001, 1023, 2473
1026  EPPR  300, 346, 363, 471, 567, 602, 647, 711
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>213</td>
<td>ESTM</td>
<td>206</td>
</tr>
<tr>
<td>175</td>
<td>EVAL</td>
<td>137, 140</td>
</tr>
<tr>
<td>131</td>
<td>EXCM</td>
<td>127</td>
</tr>
<tr>
<td>236</td>
<td>EXEC</td>
<td>216, 222</td>
</tr>
<tr>
<td>732</td>
<td>FILE</td>
<td>724</td>
</tr>
<tr>
<td>431</td>
<td>FPER</td>
<td>411</td>
</tr>
<tr>
<td>404</td>
<td>FPTR</td>
<td>172</td>
</tr>
<tr>
<td>207</td>
<td>ITIM</td>
<td>203</td>
</tr>
<tr>
<td>1024</td>
<td>LOAD</td>
<td>1006</td>
</tr>
<tr>
<td>556</td>
<td>LOGP</td>
<td></td>
</tr>
<tr>
<td>2136</td>
<td>LSTA</td>
<td>2123</td>
</tr>
<tr>
<td>663</td>
<td>LSTM</td>
<td>706, 770, 1152</td>
</tr>
<tr>
<td>1447</td>
<td>LSUB</td>
<td>133</td>
</tr>
<tr>
<td>102</td>
<td>MDST</td>
<td>60, 62, 64, 75, 100</td>
</tr>
<tr>
<td>2773</td>
<td>NAME</td>
<td>111, 123, 1420</td>
</tr>
<tr>
<td>2025</td>
<td>NREP</td>
<td>1675, 1676, 1735</td>
</tr>
<tr>
<td>36</td>
<td>NSYM</td>
<td>66, 103, 105, 111, 113, 116, 123, 1412, 1413, 1415, 1420, 2773, 3031, 3067, 3125</td>
</tr>
<tr>
<td>1260</td>
<td>OCTI</td>
<td>1301</td>
</tr>
<tr>
<td>4</td>
<td>PCTI</td>
<td>1410</td>
</tr>
<tr>
<td>1510</td>
<td>PMBP</td>
<td>1463</td>
</tr>
<tr>
<td>2625</td>
<td>P&lt;MGW</td>
<td>2613, 2622</td>
</tr>
<tr>
<td>2564</td>
<td>PRLP</td>
<td>2613, 2621, 2624</td>
</tr>
<tr>
<td>2541</td>
<td>PTXI</td>
<td>2530</td>
</tr>
<tr>
<td>261</td>
<td>QUIT</td>
<td>672, 705, 1150, 1252</td>
</tr>
<tr>
<td>610</td>
<td>SADR</td>
<td>332, 531, 540, 541, 544, 546, 547, 556, 571</td>
</tr>
<tr>
<td>361</td>
<td>SEQI</td>
<td>266, 320, 430, 530, 537</td>
</tr>
<tr>
<td>1167</td>
<td>SIND</td>
<td>1162, 1545</td>
</tr>
<tr>
<td>2406</td>
<td>SIZE</td>
<td>2060</td>
</tr>
<tr>
<td>1412</td>
<td>SLPM</td>
<td>1447</td>
</tr>
<tr>
<td>541</td>
<td>SLRL</td>
<td>333, 532</td>
</tr>
<tr>
<td>453</td>
<td>SPER</td>
<td>442, 450</td>
</tr>
<tr>
<td>2633</td>
<td>SPRA</td>
<td>2554, 2637</td>
</tr>
<tr>
<td>2660</td>
<td>SPRS</td>
<td>2553</td>
</tr>
<tr>
<td>73</td>
<td>SRCN</td>
<td>55, 56, 101, 130</td>
</tr>
<tr>
<td>723</td>
<td>SCTRL</td>
<td></td>
</tr>
<tr>
<td>314</td>
<td>STRR</td>
<td>170</td>
</tr>
<tr>
<td>1172</td>
<td>SVFQ</td>
<td>1161, 1524</td>
</tr>
<tr>
<td>2146</td>
<td>TABL</td>
<td>2063, 2125, 2407</td>
</tr>
<tr>
<td>291</td>
<td>TCAP</td>
<td>176</td>
</tr>
<tr>
<td>237</td>
<td>TIME</td>
<td>205</td>
</tr>
<tr>
<td>524</td>
<td>TIMR</td>
<td>171</td>
</tr>
<tr>
<td>133</td>
<td>TMAP</td>
<td>126</td>
</tr>
<tr>
<td>477</td>
<td>WACO</td>
<td>446</td>
</tr>
<tr>
<td>503</td>
<td>WMCO</td>
<td>452</td>
</tr>
<tr>
<td>2027</td>
<td>WREP</td>
<td>1710, 1713, 1716, 1717, 1727, 1731</td>
</tr>
<tr>
<td>2442</td>
<td>WTX4</td>
<td>2407</td>
</tr>
<tr>
<td>510</td>
<td>XFPT</td>
<td>460, 466, 470</td>
</tr>
<tr>
<td>350</td>
<td>XSTR</td>
<td>341, 343, 345</td>
</tr>
<tr>
<td>2572</td>
<td>ZONE</td>
<td></td>
</tr>
<tr>
<td>742</td>
<td>CLOSE</td>
<td>733</td>
</tr>
<tr>
<td>1256</td>
<td>COMBT</td>
<td>1243, 1241</td>
</tr>
<tr>
<td>1250</td>
<td>COMSW</td>
<td>1234, 1235, 1253</td>
</tr>
<tr>
<td>173</td>
<td>(113)</td>
<td>37</td>
</tr>
<tr>
<td>174</td>
<td>((ST)</td>
<td>41</td>
</tr>
<tr>
<td>1602</td>
<td>DGNX4</td>
<td>1523</td>
</tr>
</tbody>
</table>
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
POST PROCESSOR ASSEMBLY DATA

2026 DNREP 1677, 1704, 1706
1002 DOOR1 776, 1020
1517 ENDPM 1471
1027 ENDR1 715
1077 ENDP 746, 2457
2406 ENBL 2467, 3067
ENTRY 105, 116
714 FINIS 1473
627 IDTR 174
1660 LOOP1 1650, 1651, 1747
1657 LOOP2 1775
2116 LOWER 2061, 2070
2024 LWORD 1662, 1711, 1714
2663 NOOPC 2636
2135 OFBD 2054, 2065, 2117
2125 OFPND 2053, 2067
2022 PMCT 1431, 1440, 1451, 1460, 1721, 1726, 1751, 1756, 2005, 2012
2173 PRRCG 1407
2657 PRDCL 2564, 2571
2664 PRGCM 2634
2520 PRINT 725, 735, 763, 777, 1021, 2444, 2447, 2470, 2665
1474 PROER 1462
571 PSTOP 543, 554
2115 RAISE 2066
2612 RLECP 2574, 2605
679 RLSTM 701
1113 REXIT 1000
677 SLSTM 663, 666, 673, 675
2062 SRC1 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2110, 2111
2112
2066 SRC2 2062
1146 SVCCN 221, 264, 316, 427, 434, 526, 535, 631
701 TLSTM 667
262 TRAPR 167
2117 TYPEA 2057, 2113, 2114
2605 TEG 2577, 2602
2614 WDCNT 2540, 2543, 2617
474 WEVER 441
2435 WTCL 2433
2450 WTCOM 2426, 2431, 2434, 2441
2436 WTLSW 2430
2451 WTX 2414, 2443
2432 WTTX 2417
2432 WTXI 2416
1105 XFILE 736
651 XIORD 642, 644, 646
611 XIOP 562, 564, 566
1126 XRCAL 1022
365 XSEQR 362
1031 XSTOP 726
302 XRTRAP 273, 275, 277
1301 BCOTAB 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1320, 1321
1322, 1323, 1324, 1325, 1326, 1327, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1340, 1341, 1342
1343, 1344, 1345, 1346, 1347, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1360, 1361, 1362, 1363
1364, 1365, 1366, 1367, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1400

141
TESTS) FOR CAP, MONITOR FOR CLASS ASSEMBLY PROGRAM.
POST PROCESSOR ASSEMBLY DATA

2665 CARDIM 2560, 2561, 2576, 2601, 2603, 2607, 2611, 2627
1233 CDMAER
1604 HDAC01 1563
1610 HDAC02 1534, 1565
1613 HDAC03 1537, 1540, 1567
1617 HDAC04 1543, 1544, 1571
1623 HDAC05 1547, 1550, 1573
1627 HDAC06 1553, 1575
1632 HDAC07 1556, 1577
1635 HDAC08 1561, 1601
0 MOVIE) 52, 54
2520 NPRINT 2665
1173 OCTADR 114, 117, 223, 270, 336, 464, 557, 572, 637, 1552, 1555, 1560, 1654
1640 OCTMP 1444, 1466
3031 ORIGIN 103, 113, 1413, 1415
604 PSTOP5 952
1102 RENOPT 754
1207 TABBLK 1203, 1207, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1220, 1221, 1222, 1223, 1224, 1225, 1226
1227, 1230, 1231, 1232, 1707, 1767
7 TESTS)
2476 WDOTA3 2471
1061 XCLOSE 764
141 XMAP01 70
147 XMAP02 72
153 XMAP03 112, 115, 120, 122
157 XMAP04 132
163 XMAP05 135
621 XPSTOP 575, 577, 601
2047 (CPCD) 1712, 1740

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP.. 000054 IN HUNDREDTHS OF MINUTES.*
BUFFERED INPUT/OUTPUT PROGRAMS FOR CAP.

PCC
COUNT  278
LBL  RIP
00012  ENTRY  RIP  BINARY CARD LABEL.
00007  ENTRY  WCT1  SETUP READ OPERATIONS.
00017  ENTRY  WC1  WRITE-COLLATION-TAPE-1.
00018  ENTRY  REIND  READ COLLATION TAPE.
00023  ENTRY  READ2  REIND.
00024  ENTRY  PRINT  BUFFERED PRINT PROGRAM.
00027  ENTRY  PPROG  DUMP SYMBOCL PROGRAM.
00027  ENTRY  PCT1  DUMP COLLATION TAPE.
00027  ENTRY  READ1  READ INPUT TAPE.

RIP SETS UP READI AND PPROG.

TRANSFER VECTOR
00000  475146276060  PROG
00001  626704606060  SX4
00002  626523464060  SVCSCON
00003  664663606060  WGT
00004  254744516060  EPMR
00005  222123426060  BACK
00006  436263446060  LSTM
00007  222324632122  BCDTAB

LINKAGE DIRECTOR
00010  000000000060
00011  513147606060
00012  0634  00  0  00122  RIP  SXA  RX+4  SAVE IR+4.
00013  -0500  60  0  00000  CAL*  $PROG  GET CONTROL WORD.
00014  -0734  00  4  00000  PDX  0.4  COUNT TC IR+4.
00015  -3  04064  4  00017  TXL  **2,4,LPROG  MAXIMUM NUMBER OF CARDS IN PROG.
00016  0774  00  4  04064  AXI  LPROG,4  **
00017  0754  00  4  00000  PXX  0.4  COUNT TO A(AC).
00020  0361  60  0  00000  ACL*  $PROG  GET END ADDRESS.
00021  0621  00  0  00040  STA  PEND  SAVE IN READ1.
00022  0621  00  0  00312  STA  PPEND  SAVE IN PPROG.
00023  0634  00  4  00037  SXA  PLTH,4  SAVE IN READ1.
00024  0634  00  4  00310  SXA  PPLTH,4  SAVE IN PPROG.
00025  0534  00  4  00122  LXG  RX+4  RESTORE IR+4.
00026  0020  00  0  00001  TRA  14  RETURN.

SPACE  2
READI TAKES NEXT 14 WORDS OF PROG AND STORES IN BUFF.

00027  0520  00  0  00057  READI  ZET  TECFI  CHECK FOR END OF PROGRAM.
00030  0020  00  0  00051  TRA  EOFI  END REACHED, COMMENT.
00031  0634  00  4  00122  SXA  RX+4  SAVE IRS.
00032  0634  00  2  00123  SXA  RX+2  **
00033  -0500  60  0  00001  CAL  1.4  GET CONTROL WORD.
00034  0361  00  0  10611  ACL  =14  COMPUTE BUFFER+14.
00035  0621  00  0  00041  STA  SLW1  INSERT.
00036  0774  00  4  00016  AXI  14,4  SET WORD COUNT.
00037  0774  00  2  00000  PLTH  AXI  **2  SET PROGRAM INDEX.
### BUFFERED INPUT/OUTPUT PROGRAMS FOR CAP.

<table>
<thead>
<tr>
<th>Address</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00040-00041</td>
<td>PEND CAL **,2</td>
<td>COPY CARD INTO BUFFER.</td>
</tr>
<tr>
<td>00042-00043</td>
<td>SLW SLW **,4</td>
<td>**</td>
</tr>
<tr>
<td>00044-00045</td>
<td>TIX **2,2,1</td>
<td>IF END CF PROG, SET TEOF.</td>
</tr>
<tr>
<td>00046-00047</td>
<td>STL TEOF</td>
<td>**</td>
</tr>
<tr>
<td>00048-00049</td>
<td>TIX PEND,4,1</td>
<td>COUNT WORDS.</td>
</tr>
<tr>
<td>00050-00051</td>
<td>STL PLTH,2</td>
<td>SAVE PROGRAM INDEX.</td>
</tr>
<tr>
<td>00052-00053</td>
<td>TIX RX,4,4</td>
<td>RESTORE IRS.</td>
</tr>
<tr>
<td>00054-00055</td>
<td>LXA TX,2,2</td>
<td>RETURN.</td>
</tr>
<tr>
<td>00056-00057</td>
<td>TRA 2,4</td>
<td></td>
</tr>
<tr>
<td>00058-00059</td>
<td>EDOF XEC*</td>
<td>$S\times$</td>
</tr>
<tr>
<td>00060-00061</td>
<td>EDT XEC*</td>
<td>SAVE IR$.</td>
</tr>
<tr>
<td>00062-00063</td>
<td>EDT XEC*</td>
<td>SAVE CONSOLE.</td>
</tr>
<tr>
<td>00064-00065</td>
<td>EDT XEC*</td>
<td>COMMENT, EDOF ON INPUT.</td>
</tr>
<tr>
<td>00066-00067</td>
<td>EDT XEC*</td>
<td>SET ERR(OR) INDICATOR.</td>
</tr>
<tr>
<td>00068-00069</td>
<td>EDT XEC*</td>
<td>EXIT TO POST MORM.</td>
</tr>
<tr>
<td>00070-00071</td>
<td>EDT XEC*</td>
<td>END OF FILE MARK ON INPUT TAPE.</td>
</tr>
<tr>
<td>00072-00073</td>
<td>EDT XEC*</td>
<td>WEDOF I PZ.</td>
</tr>
<tr>
<td>00074-00075</td>
<td>EDT XEC*</td>
<td>WEDOF I BC</td>
</tr>
<tr>
<td>00076-00077</td>
<td>EDT XEC*</td>
<td>END OF FILE REACHED WHILE READING CAP INPUT TAPE.</td>
</tr>
</tbody>
</table>

---

### SPACE 2

WCT1 TAKES 14 WORDS AND WRITES THEM ON COLLATION TAPE BUFFER. IF TAG OF CONTROL WORD IS NON-ZERO, THEN N (IN DECREMENT OF CONTROL WORD) WORDS OF BUFFER AND 14-N BLANK WORDS GO TO COLLATION TAPE BUFFER.

---

<table>
<thead>
<tr>
<th>Address</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00078</td>
<td>PEND CAL **,2</td>
<td></td>
</tr>
<tr>
<td>00079</td>
<td>SLW SLW **,4</td>
<td>**</td>
</tr>
<tr>
<td>00080</td>
<td>TIX **2,2,1</td>
<td>IF END CF PROG, SET TEOF.</td>
</tr>
<tr>
<td>00081</td>
<td>STL TEOF</td>
<td>**</td>
</tr>
<tr>
<td>00082-00083</td>
<td>TIX PEND,4,1</td>
<td>COUNT WORDS.</td>
</tr>
<tr>
<td>00084-00085</td>
<td>STL PLTH,2</td>
<td>SAVE PROGRAM INDEX.</td>
</tr>
<tr>
<td>00086-00087</td>
<td>TIX RX,4,4</td>
<td>RESTORE IRS.</td>
</tr>
<tr>
<td>00088-00089</td>
<td>LXA TX,2,2</td>
<td>RETURN.</td>
</tr>
<tr>
<td>00090-00091</td>
<td>TRA 2,4</td>
<td></td>
</tr>
<tr>
<td>00092-00093</td>
<td>EDOF XEC*</td>
<td>$S\times$</td>
</tr>
<tr>
<td>00094-00095</td>
<td>EDT XEC*</td>
<td>SAVE IR$.</td>
</tr>
<tr>
<td>00096-00097</td>
<td>EDT XEC*</td>
<td>SAVE CONSOLE.</td>
</tr>
<tr>
<td>00098-00099</td>
<td>EDT XEC*</td>
<td>COMMENT, EDOF ON INPUT.</td>
</tr>
<tr>
<td>00100-00101</td>
<td>EDT XEC*</td>
<td>SET ERR(OR) INDICATOR.</td>
</tr>
<tr>
<td>00102-00103</td>
<td>EDT XEC*</td>
<td>EXIT TO POST MORM.</td>
</tr>
<tr>
<td>00104-00105</td>
<td>EDT XEC*</td>
<td>END OF FILE MARK ON INPUT TAPE.</td>
</tr>
<tr>
<td>00106-00107</td>
<td>EDT XEC*</td>
<td>WEDOF I PZ.</td>
</tr>
<tr>
<td>00108-00109</td>
<td>EDT XEC*</td>
<td>WEDOF I BC</td>
</tr>
<tr>
<td>00110-00111</td>
<td>EDT XEC*</td>
<td>END OF FILE REACHED WHILE READING CAP INPUT TAPE.</td>
</tr>
</tbody>
</table>
BUFFERED INPUT/OUTPUT PROGRAMS FOR CAP.

00115 0602 00 1 10571 SLW CT1,1 WCT. RIP00870
00116 2 00001 1 00120 TIX **2.1,1 COUNT WORDS CT1. RIP00880
00117 -0625 00 0 00135 STL TECTC END OF CT1, SET TECTC. RIP00890
00120 2 00001 4 00112 TIX WCAL,4,1 COUNT WORDS IN RECORD. RIP00900
00121 0634 00 1 00111 SXA WLCT,1 SAVE CT1 INDEX. RIP00910
00122 0774 00 4 00000 RX4 AXT **,4 RESTORE IRs. RIP00920
00123 0774 00 2 00000 RX2 AXT **,2 -- RIP00930
00124 0774 00 1 00000 RX1 AXT **,1 -- RIP00940
00125 0020 00 4 00002 TRA 2,1 RETURN. RIP00950

00126 0522 60 0 00001 EOTC XEC= X$X4 SAVE IR4. RIP00960
00127 0774 00 4 00002 TSX $SVC0N,4 SAVE CONSOLE. RIP00970
00130 0774 00 4 00003 TSX $SOT,4 COMMENT, EOT CT1. RIP00980
00131 0 00011 0 00136 PZE WECTC,0,9 -- RIP00990
00132 -0625 60 0 00004 STL+ SPEMR SET ERROR INDICATOR. RIP01000
00133 0020 60 0 00005 TRA+ $BACK EXIT TO POST MORTEM. RIP01010

00134 0 00000 0 00000 TAG PZE CONTROL WORD FOR WCT.
00135 0 00000 0 00000 TECTC PZE END OF TAPE MARK ON COLLATION TAPE.
00136 002545246046 WECTC BCI 9,0 END OF TAPE REACHED WHILE WRITING CAP COLLATION TAPE. RIP01050
00137 26663214725
00140 265125212330
00141 252460663031
00142 432560665131
00143 63314527023
00144 214760234643
00145 432163146445
00146 606321472533

SPACE 2 RIP01070
REWIND SETS UP READ2 AND PCT1. READ2 AND PCT1 CHECK THAT REWIND HAS BEEN CALLED.

00147 -0625 00 0 00156 REWIND STL TREW CT1 REWOUND. RIP01080
00150 -0500 00 0 00111 CAL WLCT GET CT1 INDEX IN A(AC). RIP01090
00151 0767 00 0 00022 ALS 18 PLACE IN D(AC). RIP01100
00152 0622 00 0 00175 STU TLCT SAVE IN READ2. RIP01110
00153 0622 00 0 00403 STG PTLCT SAVE IN PCT1. RIP01120
00154 0140 00 0 00155 TOV **,1 TURN OFF A(OC). RIP01130
00155 0200 00 4 00001 TRA 1,1 RETURN. RIP01140

00156 0 00000 0 00000 TREW PZE BEGINNING OF TAPE MARK ON COLLATION TAPE. RIP01150

SPACE 2 RIP01160
READ2 READS ONE RECORD OF THE COLLATION TAPE BUFFER INTO BUFFER OF CONTROL WORD.

00157 0634 00 4 00122 REAC2 SXA RX4,4 SAVE IR5. RIP01170
00160 0634 00 2 00123 SXA RX2,2 -- RIP01180
00161 -0520 00 0 00156 NZT TREW TEST FOR REWIND. RIP01190
00162 0200 00 0 00204 TRA NREW NOT REWOUND, COMMENT. RIP01200
00163 0520 00 0 00231 ZET TEDFC CHECK FOR EOF ON CT1. RIP01210
00164 0200 00 0 00223 TRA EDFC EOF REACHED, COMMENT. RIP01220
00165 0774 00 2 10150 RLCT AXT LCT1,2 MAXIMUM NUMBER OF CARDS ON CT1. RIP01230

145
<table>
<thead>
<tr>
<th>Address</th>
<th>Assembly Code</th>
<th>Comment</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>00166</td>
<td>-0500 00 4 00001</td>
<td>CAL 1,4 GET CONTROL WORD.</td>
<td>R1P01310</td>
</tr>
<tr>
<td>00167</td>
<td>0361 00 0 10611</td>
<td>AGL =14 COMPUTE BUFFER+14.</td>
<td>R1P01320</td>
</tr>
<tr>
<td>00170</td>
<td>0621 00 0 00173</td>
<td>STA **3 INSERT.</td>
<td>R1P01330</td>
</tr>
<tr>
<td>00171</td>
<td>0774 00 4 00016</td>
<td>AXT 14,4 SET WORD COUNT.</td>
<td>R1P01340</td>
</tr>
<tr>
<td>00172</td>
<td>-0500 00 2 10571</td>
<td>CAL CT1,2 MOVE TO BUFFER.</td>
<td>R1P01350</td>
</tr>
<tr>
<td>00173</td>
<td>0602 00 4 00000</td>
<td>SLW **4 **.</td>
<td>R1P01360</td>
</tr>
<tr>
<td>00174</td>
<td>1 7777 2 00175</td>
<td>TXI **1,2,-1 COUNT CT1.</td>
<td>R1P01370</td>
</tr>
<tr>
<td>00175</td>
<td>3 00000 2 00177</td>
<td>TLCT TXH <strong>2,2,</strong> CHECK FOR FILE MARK.</td>
<td>R1P01380</td>
</tr>
<tr>
<td>00176</td>
<td>-0625 00 0 00231</td>
<td>STL TECFC FILE MARK.</td>
<td>R1P01390</td>
</tr>
<tr>
<td>00177</td>
<td>2 00001 4 00172</td>
<td>TIX +5,4,1 COUNT WCRDS.</td>
<td>R1P01400</td>
</tr>
<tr>
<td>00200</td>
<td>0634 00 2 00016</td>
<td>SXA RLCT,2 SAVE CTI INDEX.</td>
<td>R1P01410</td>
</tr>
<tr>
<td>00201</td>
<td>0534 00 4 00122</td>
<td>LXA RXA,4 RESTORE IRS.</td>
<td>R1P01420</td>
</tr>
<tr>
<td>00202</td>
<td>0534 00 2 00123</td>
<td>LXA RXA,2 **.</td>
<td>R1P01430</td>
</tr>
<tr>
<td>00203</td>
<td>0200 00 4 00002</td>
<td>TRA 2,4 RETURN.</td>
<td>R1P01440</td>
</tr>
<tr>
<td>00204</td>
<td>0074 00 4 00006</td>
<td>NREW TSX $LSIM,4 CTI NOT REWOUND, RESET TRAP.</td>
<td>R1P01450</td>
</tr>
<tr>
<td>00205</td>
<td>0074 00 4 00003</td>
<td>TSX $WOT,4 COMMENT.</td>
<td>R1P01460</td>
</tr>
<tr>
<td>00206</td>
<td>0 00011 0 00212</td>
<td>PZE WNREM,0,9 **.</td>
<td>R1P01470</td>
</tr>
<tr>
<td>00207</td>
<td>-0760 00 0 0005</td>
<td>ESTM RE-ENTER TRAP.</td>
<td>R1P01480</td>
</tr>
<tr>
<td>00210</td>
<td>0074 00 4 00147</td>
<td>TSX REWIND,4 REMIND CTI.</td>
<td>R1P01490</td>
</tr>
<tr>
<td>00211</td>
<td>0020 00 0 00165</td>
<td>TRA RLCT RETURN.</td>
<td>R1P01500</td>
</tr>
<tr>
<td>00212</td>
<td>0051252172402</td>
<td>WNWEM BCI 9,CREAD2 CALLED BEFORE REWIND, COLLATION TAPE REWOUND.</td>
<td>R1P01510</td>
</tr>
<tr>
<td>00213</td>
<td>602321434325</td>
<td>CREAD2</td>
<td>R1P01520</td>
</tr>
<tr>
<td>00214</td>
<td>246022252646</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00215</td>
<td>512560512566</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00216</td>
<td>314524736023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00217</td>
<td>46443216331</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00220</td>
<td>464560632147</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00221</td>
<td>256051256664</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00222</td>
<td>444524336060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00223</td>
<td>0522 60 0 00001</td>
<td>EDFC XEC $SX4 SAVE IR4.</td>
<td>R1P01530</td>
</tr>
<tr>
<td>00224</td>
<td>0074 00 4 00002</td>
<td>TSX $WCON,4 SAVE CONSOLE.</td>
<td>R1P01540</td>
</tr>
<tr>
<td>00225</td>
<td>0074 00 4 00003</td>
<td>TSX $WOT,4 COMMENT, EOF ON CTI.</td>
<td>R1P01550</td>
</tr>
<tr>
<td>00226</td>
<td>0 00111 0 00232</td>
<td>PZE WEFCC,0,9 **.</td>
<td>R1P01560</td>
</tr>
<tr>
<td>00227</td>
<td>-0625 60 0 00004</td>
<td>STL* $EMP SET ERROR INDICATOR.</td>
<td>R1P01570</td>
</tr>
<tr>
<td>00230</td>
<td>0020 60 0 00005</td>
<td>TRA* $BACK EXIT TO POST MORTEM.</td>
<td>R1P01580</td>
</tr>
<tr>
<td>00231</td>
<td>0 00000 0 00000</td>
<td>TEOFC PZE END OF FILE MARK ON COLLATION TAPE.</td>
<td>R1P01590</td>
</tr>
<tr>
<td>00232</td>
<td>002545246046</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00233</td>
<td>2660226314329</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00234</td>
<td>605125212330</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00235</td>
<td>252460663031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00236</td>
<td>432560512521</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00237</td>
<td>243145276023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00240</td>
<td>464343216331</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00241</td>
<td>464560632147</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00242</td>
<td>253360606060</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SPACE** 2

PRINT COUNTS LINES AND CALLS $WOT.  

<table>
<thead>
<tr>
<th>Address</th>
<th>Assembly Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00243</td>
<td>0634 00 4 00122</td>
<td>PRINT SXA RX4,4 SAVE IR4.</td>
</tr>
<tr>
<td>00244</td>
<td>0500 00 0 00267</td>
<td>CLA LNCNT COUNT LINES.</td>
</tr>
</tbody>
</table>
BUFFED INPUT/OUTPUT PROGRAMS FOR CAP.

00326 -0600 00 0 10571
00327 0074 00 4 00003
00330 0 0017 0 10571
00331 2 00011 1 00311
00332 0534 00 4 00122
00333 0534 00 2 00123
00334 0534 00 1 00124
00335 0200 00 4 00001
00336 0 00070 0 00072
00337 016060606060
00340 60660606060
00341 60474626360
00342 44465163544
00343 604626606270
00344 442246433123
00345 604751462751
00346 21443606060

SPACE 2

PCT1 PRINTS UP TO 300 RECORDS OF CT1.

00347 0634 00 4 00122
00350 0634 00 2 00123
00351 0634 00 1 00124
00352 0074 00 4 00003
00353 0 0010 0 00411
00354 074 00 4 00003
00355 00001 0 10614
00356 -0500 00 0 00007
00357 0621 00 0 0365
00360 -0520 00 0 00156
00361 074 00 4 00147
00362 074 00 1 10150
00363 074 00 2 00116
00364 0600 00 1 10571
00365 -0154 00 0 00000
00366 -0600 00 2 10610
00367 1 7777 1 00370
00370 2 00011 2 00364
00371 0560 00 0 10614
00372 0500 00 0 00410
00373 0402 00 0 10610
00374 0120 00 0 00377
00375 0500 00 0 10612
00376 0560 00 0 10613
00377 0601 00 0 00410
00400 -0600 00 0 10571
00401 0074 00 4 00003
00402 0 0017 0 10571
00403 3 00001 0 00360
00404 0534 00 4 00122
00405 0534 00 2 00123
00406 0534 00 1 00124
00407 0200 00 4 00001

PTLC

00328 STG PBUFF INSERT CARRIAGE CONTROL.
00329 TSX $MCT,4 PRINT THIS CARD.
00330 PZE PBUFF,0,15 ** COUNT RECORDS.
00331 TIX PPLTH+1,1,1 RESTORE IRS.
00332 LXA RX4,4 ** RETURN.
00333 LXA RX2,2 **
00334 LXA RX1,1 **
00335 TRA 1,4 RETURN.
00336 PPCT PZE 58 60 LINES PER PAGE - 2 FOR HEADING.
00337 PHEAD BCI 2,1
00338 BCI 6, POST MCRTM OF SYMBOLIC PROGRAM.

PCT1

00347 SXA RX4,4 SAVE IRS.
00350 SXA RX2,2 **
00351 SXA RX1,1 **
00352 TSX $WOT,4 INITIAL COMMENT.
00353 PZE $HEAD,0,8 **
00354 TSX $WOT,4 BLANK LINE.
00355 PZE *H **,G,1 **
00356 CAL $BCTAB GET ADDRESS OF TABLE.
00357 STA CRC2 **
00358 STA TREW TEST FOR REWIN.
00359 STA RCW **
00360 STA CT1,1 SET MAXIMUM RECORD COUNT, COLLATION TAPE.
00361 AXT AXT 14,2 BUFFER WORD COUNT.
00362 AXT CT1,1 GET CURRENT WORD.
00363 AXT CT1,1 DELETE ILLEGAL CHARACTERS.
00364 CRQ2 CTQ **,0,6 PLACE IN BUFFER.
00365 SRT PBUFF+15,2 COUNT WORDS ON CT1.
00366 -0600 00 2 10610 COUNT WORDS IN BUFFER.
00367 1 7777 1 00370 PICKUP STANDARD CARRIAGE CONTROL.
00370 2 00011 2 00364 COUNT WORDS IN BUFFER.
00371 0560 00 0 10614 COUNT WORDS IN BUFFER.
00372 0500 00 0 00410 COUNT WORDS IN BUFFER.
00373 0402 00 0 10610 COUNT WORDS IN BUFFER.
00374 0120 00 0 00377 COUNT WORDS IN BUFFER.
00375 0500 00 0 10612 COUNT WORDS IN BUFFER.
00376 0560 00 0 10613 COUNT WORDS IN BUFFER.
00377 0601 00 0 00410 COUNT WORDS IN BUFFER.
00400 -0600 00 0 10571 COUNT WORDS IN BUFFER.
00401 0074 00 4 00003 PRINT THIS RECORD.
00402 0 0017 0 10571 PRINT THIS RECORD.
00403 3 00001 0 00360 PRINT THIS RECORD.
00404 0534 00 4 00122 PRINT THIS RECORD.
00405 0534 00 2 00123 PRINT THIS RECORD.
00406 0534 00 1 00124 PRINT THIS RECORD.
00407 0200 00 4 00001 PRINT THIS RECORD.

RIPO2170
RIPO2180
RIPO2190
RIPO2200
RIPO2210
RIPO2220
RIPO2230
RIPO2240
RIPO2250
RIPO2260
RIPO2270
RIPO2280
RIPO2290
RIPO2300
RIPO2310
RIPO2320
RIPO2330
RIPO2340
RIPO2350
RIPO2360
RIPO2370
RIPO2380
RIPO2390
RIPO2400
RIPO2410
RIPO2420
RIPO2430
RIPO2440
RIPO2450
RIPO2460
RIPO2470
RIPO2480
RIPO2490
RIPO2500
RIPO2510
RIPO2520
RIPO2530
RIPO2540
RIPO2550
RIPO2560
RIPO2570
RIPO2580
RIPO2590
RIPO2600
RIPO2610
RIPO2620
RIPO2630
RIPO2640
BUFFERED INPUT/OUTPUT PROGRAMS FOR CAP.

00410 0 00000 0 00072  PGCNT PZE  58  60 LINES PER PAGE - 2 FOR HEADING.  RIP02650
00411 016060606060   CHEAD BCI  2,1  RIP02660
00412 606060606060   BCI  6, POST MORTEM OF COLLATION TAPE.  RIP02670
00413 604966666660  4446561632544  RIP02680
00415 604626002346  44462602346  RIP02690
00416 434321633146  4446063214725  RIP02700
00417 336060606060  04064  LPROG EQU  14*150  150 CARDS ON INPUT TAPE.  RIP02710
01010  LCT1 EQU  14*300  300 RECORDS MAX CN COLLATION TAPE.  RIP02720
00420 152523465124  BCI  9, RECORD NO. 1, COLLATION TAPE, NOTHING HAS BEEN WRITTEN.  RIP02730
00421 04546336001  00422 216331464560  BCI  5, ON CTL.  RIP02740
00423 736023464343  00424 632147257360  RIP02750
00425 454663303145  00426 760330216260  RIP02760
00427 2227525456066  00428 513163632545  RIP02770
00429 00430 604649602363  00431 606060606060  RIP02780
00432 00433 13360606060  00434 606060606060  END
00435 606060606060  00436 606060606060  RIP02790
10571  CTP BES LCT1-14  COLLATION TAPE BUFFER.  RIP02800
10571  PBUFF BSS  15  PRINT BUFFER.  RIP02810
10610 0000000000001  RIP02820
10611 000000000016  RIP02830
10612 000000000073  RIP02840
10613 016060606060  RIP02850
10614 606060606060  RIP02860
REFERENCES TO DEFINED SYMBOLS
10571  CTI  115, 172, 364
12  RIP
124  RX1  75, 301, 334, 351, 406
123  RX2  32, 47, 74, 160, 202, 300, 333, 350, 405
122  RX4  12, 25, 31, 46, 73, 157, 201, 243, 257, 277, 332, 347, 404
1  SX4  51, 120, 223, 261
134  TAG  77, 101
3  WDT  53, 130, 205, 225, 254, 263, 302, 304, 327, 352, 354, 401
5  BACK  56, 133, 230, 266
313  CRCL  307
365  CRDL  357
255  CWCT  251, 252
223  EDFC  164
51  EDIF  30
126  EDTG  72
4  EPMR  55, 132, 227, 265
10150  LCTI  111, 165, 362, 421, 10571
261  LNEX  247
6  LSTM  204, 253
204  NREM  162
347  PCTI
363  PCTL  403
40  PEND  21, 44
37  PLTH  23, 45
0  PREG  13, 20
165  RLCT  200, 211
41  SLWI  35
175  TLCT  152
156  TREW  147, 161, 360
112  WCAL  167, 120
71  WCMI  111
11  WLCT  121, 150
411  CHEAD  353
267  LNCNT  244, 246
4064  LPRG  15, 16, 421
10571  PRBFF  314, 326, 330, 366, 400, 402
410  PCCNT  372, 377
337  PHEAD  303
336  PPCNT  320, 325
312  PPEND  22
310  PLLTH  24, 331
277  PPREG  243
243  PRINT
403  PTLCT  153
27  READI
157  READD  2
2  SVCCN  52, 127, 224, 262
231  TEDFC  163, 176
57  TEDFI  27, 43
135  TEDTC  71, 117
232  WEDFC  226
60  WEDFI  54
BUFFERED INPUT/OUTPUT PROGRAMS FOR CAP.
POST PROCESSOR ASSEMBLY DATA

136  WEOIC  131
270  WLNX   264
212  WNREW  206
7    BCDTAB  306,  356
147  REWIND  210,  361

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP... 000012 IN HUNDREDTHS OF MINUTES.
SYMBOLIC PROGRAM TO TEST CAP.

**I,0,LTH**  
**+I,0,LTH**  
**-I,0,LTH**  
**0,0,LTH**

PCC  150  BINARY CARD LABEL.

00002  PRG00010
00003  PRG00020
00004  PRG00030
00005  PRG00040
00006  PRG00050

**TITLE**  
**DEST**  
**CNT**

**ENTRY**  
**PRG**

**00000**  
**00001**

**PROG**  
**PZE**

00003  9,CAP  REM THE FOLLOWING ARE ALL LEGAL CAP INSTRUCTIONS  
        9,NCAP  
        9,NCAP  
45623360660  BCI  

00014  5NS  REM PROGRAM TO COUNT BITS IN AC.

00021  606060606600  BCI  

00032  606060606600  BCI  

00037  234666456360  BCI  

00050  606060606600  BCI  

00055  606060606600  BCI  

00066  606060606600  BCI  

00073  606060606600  BCI  

00104  606060606600  BCI  

00111  43466476680  BCI  

00112  606060606600  BCI  

00127  606060606600  BCI  

00140  606060606600  BCI  

00145  702562606600  BCI  

00156  606060606600  BCI  

00163  606060606600  BCI  

00174  606060606600  BCI  

00201  606060606600  BCI  

00212  606060606600  BCI  

00217  606060606600  BCI  

00230  606060606600  BCI  

00235  606060606600  BCI  

00246  606060606600  BCI  

00253  606060606600  BCI  

00264  606060606600  BCI  

00271  454660606600  BCI  

00302  606060606600  BCI  

00307  606060606600  BCI  

00320  606060606600  BCI  

00325  244652566060  BCI  

00336  606063236243  BCI  

00343  606060606600  BCI  

00354  606060606600  BCI  

00361  712551466600  BCI  

00372  606060606600  BCI  

00377  464526566600  BCI  

00410  606060606600  BCI  

00415  633062676600  BCI  

00426  606060606600  BCI  

00433  606060606600  BCI  

00444  606060606600  BCI  

00451  223163626060  BCI  

00462  606060606600  BCI  

00467  664651246600  BCI

00002 01720 00003  PROG  **PZE**  

**CORTROL WORD. START**  
**LENGTH**  
**DELETE GENERATED OCTAL FROM LISTING.**  
**PROG00060**  

**PROG00070**  

**PROG00080**  

**PROG00090**  

**PROG00100**  

**PROG00110**  

**PROG00120**  

**PROG00130**  

**PROG00140**  

**PROG00150**  

**PROG00160**  

**PROG00170**  

**PROG00180**  

**PROG00190**  

**PROG00200**  

**PROG00210**  

**PROG00220**  

**PROG00230**  

**PROG00240**  

**PROG00250**  

**PROG00260**  

**PROG00270**  

**PROG00280**  

**PROG00290**  

**PROG00300**  

**PROG00310**  

**PROG00320**  

**PROG00330**  

**PROG00340**  

**PROG00350**  

**PROG00360**  

**PROG00370**  

**PROG00380**  

**PROG00390**  

**PROG00400**  

**PROG00410**  

**PROG00420**  

**PROG00430**  

**PROG00440**  

**PROG00450**  

**PROG00460**  

**PROG00470**  

**PROG00480**  

**PROG00490**  

**PROG00500**  

**PROG00510**  

**PROG00520**
SYMBOLIC PROGRAM TO TEST CAP.

BCI 5,  TSTCAP22  PROG0530
BCI 9,  REM TEST OF CAP PSEUDO-OPS. AND FLAGS.  PROG0540
BCI 5,  TSTCAP23  PROG0550
BCI 9,  ILLEGAL OPCODE.  PROG0560
BCI 5,  TSTCAP24  PROG0570
BCI 9,  TRA UNDEF UNDEFINED SYMBOL.  PROG0580
BCI 5,  TSTCAP25  PROG0590
BCI 9,  INT 1, 2, -7, 1333, 9 ERROR INTOP.  PROG0600
BCI 5,  TSTCAP26  PROG0610
BCI 9,  WMW AEN ILLEGAL OPCODE AND UNDEFINED  PROG0620
BCI 5,  C SYMBOL. TSTCAP27  PROG0630
BCI 9,  COMP NO = YES + LOOP  PROG0640
BCI 5,  TSTCAP28  PROG0650
BCI 9,  COMP COMP Z = A + B + C / D / E / FLAG  PROG0660
BCI 5,  TSTCAP29  PROG0670
BCI 9,  TRA COMP USE OF SYMBOL DEFINED BY CD  PROG0680
BCI 5,  TSTCAP30  PROG0690
BCI 9,  COMP COUNT = LOOP01+2 - START,4  PROG0700
BCI 5,  TSTCAP31  PROG0710
BCI 9,  COMP DZ = (Z+Z*[(Z+Z*(Z+Z)*Z)]) + FOO  PROG0720
BCI 5,  TSTCAP32  PROG0730
BCI 9,  TIX COUNT = ((Z+Z*(Z+Z)*Z)+1)*(Z+Z)*Z+Z  PROG0740
BCI 5,  TSTCAP33  PROG0750
BCI 9,  REM TEST OF PROPOSED MODIFICATIONS TO CAP.  PROG0760
BCI 5,  TSTCAP34  PROG0770
BCI 9,  MUL INT 0 MULTIPLY DEFINED SYMBOL.  PROG0780
BCI 5,  TSTCAP35  PROG0790
BCI 9,  MUL INT 0 MULTIPLY DEFINED SYMBOL.  PROG0800
BCI 5,  TSTCAP36  PROG0810
BCI 9,  C PZE -32+BITS PZE CODE.  PROG0820
BCI 5,  TSTCAP37  PROG0830
BCI 9,  D MZE NO+65 MZE CODE.  PROG0840
BCI 5,  TSTCAP38  PROG0850
BCI 9,  SLW $+1 $ FOR THIS LOCATION  PROG0860
BCI 5,  TSTCAP39  PROG0870
BCI 9,  CLA 2*S-1 $ TEST.  PROG0880
BCI 5,  TSTCAP40  PROG0890
BCI 9,  MTH C/3 DIVISION IN ADDRESS  PROG0900
BCI 5,  TSTCAP41  PROG0910
BCI 9,  Z PON (YES-COUNT)/3+F PON WITH DIVISION IN ADDR  PROG0920
BCI 9,  ESS ARITHMETIC. TSTCAP42  PROG0930
BCI 9,  STO IC-FLAGS/Y2 DIVISION WITH NEGATIVE ANS  PROG0940
BCI 5,  TSTCAP43  PROG0950
BCI 9,  ER.  PROG0960
BCI 5,  TSTCAP44  PROG0970
BCI 9,  SLW -8+BITS+$+1+T+Z USE OF $ AS A SYMBOL.  PROG0980
BCI 5,  TSTCAP45  PROG0990
BCI 9,  ARF (S-C)+S, AND ILLEGAL OPCODE  PROG1000
BCI 5,  TSTCAP46  PROG1010
BCI 9,  REMARK CARD WITH * IN COLUMN 1.  PROG1020
BCI 5,  TSTCAP47  PROG1030
BCI 9,  BLNK BLNK BLANK OPCODE.  PROG1040
BCI 5,  TSTCAP48  PROG1050
BCI 9,  Q =O= MOL 5 THIS IS MOLLERITH INFO.  PROG1060
BCI 5,  TSTCAP49  PROG1070
BCI 9,  CLA* FLAG USE OF FLAGGED INSTRUCTION.  PROG1080
SYMBOLIC PROGRAM TO TEST CAP.

01310 606060606060  BCI 5,  TSTCAP50  LITERAL.
01315 606060606060  BCI 9,FLAGS  TSTCAP51
01326 606060606060  BCI 5,  CLA =1139  TSTCAP52
01333 606060606060  BCI 9,  LAC =5597.0  E, S, AND F FLAGS.
01344 606060606060  BCI 5,  TNC X,3,5  TSTCAP53
01351 606060606060  BCI 5,  TNC Y,(YES-COUNT)  TSTCAP54
01362 606060606060  BCI 5,  (NO-LOOP)+START,2+3  TSTCAP55
01367 606060606060  BCI 5,  OCT 17,13-44,0.13,,1  TSTCAP56
01400 606060606060  BCI 5,  A GCT PSEUDO-OP.
01405 606060606060  BCI 5,  FDP FLAG  TSTCAP57
01416 606060606060  BCI 9,  DIVIDE INSTRUCTION.
01423 606060606060  BCI 5,  STO MUL*UNDEF  TSTCAP58
01441 606060606060  BCI 5,  FOLLOWED BY A STORE.
01444 606060606060  BCI 9,  MULTIPLICATION.
01451 606060606060  BCI 5,  START EQU COUNT+3*(NO-LOOP)  TSTCAP59
01452 606060606060  BCI 5,  PROPER USE OF EQU.
01457 606060606060  BCI 5,  TNC X,3,5  TSTCAP60
01470 606060606060  BCI 5,  USE OF SYMBOL DEFINED BY EQ.
01475 626321516360  BCI 5,  TNC X,3,5  TSTCAP61
01477 606060606060  BCI 5,  PHASE ERROR.
01506 606060606060  BCI 9,  TRA START  TSTCAP62
01513 606060606060  BCI 5,  USE OF SYMBOL WITH PHASE ERROR.
01524 606060606060  BCI 5,  TNC X,3,5  TSTCAP63
01531 606060606060  BCI 5,  PROPER BSS.
01542 606060606060  BCI 5,  TNC X,3,5  TSTCAP64
01547 606060606060  BCI 5,  PROPER SYMBOLIC DEFINITION.
01558 606060606060  BCI 5,  BSS LOOP-COUNT  TSTCAP65
01560 594651336060  BCI 5,  IMPROPER BSS, PHASE ERROR.
01565 606060606060  BCI 5,  TNC X,3,5  TSTCAP66
01576 606060606060  BCI 5,  IMPROPER BSS, PHASE ERROR.
01603 606060606060  BCI 9,  OF BSS.
01614 606060606060  BCI 5,  TNC X,3,5  TSTCAP67
01621 606060606060  BCI 5,  IMPROPER BSS, PHASE ERROR.
01632 606060606060  BCI 5,  TNC X,3,5  TSTCAP68
01637 606060606060  BCI 5,  IMPROPER BSS, PHASE ERROR.
01650 606060606060  BCI 9,  CALL CALL COUNT,NOVOTES LEGAL CALL MACRO.
01655 606060606060  BCI 5,  TNC X,3,5  TSTCAP69
01666 606060606060  BCI 9,  CALL ABLE,BAKER+2,CHRLY=5*(NO-LOOP)
01666 606060606060  BCI 5,  TNC X,3,5  TSTCAP70
01704 433360606060  BCI 5,  CHECK ILC AFTER BSS AND CAL.
01711 602545246660  BCI 9,  TNC X,3,5  TSTCAP71
01722 606060606060  BCI 5,  RETURN TO NORMAL MODE.
01724  LTH DETAIL  PROG1090
01724  EQU LTH  PROG1100
01724  =-1-PROG  PROG1110
01724  END  PROG1120
01724  PROG1130
01724  PROG1140
01724  PROG1150
01724  PROG1160
01724  PROG1170
01724  PROG1180
01724  PROG1190
01724  PROG1200
01724  PROG1210
01724  PROG1220
01724  PROG1230
01724  PROG1240
01724  PROG1250
01724  PROG1260
01724  PROG1270
01724  PROG1280
01724  PROG1290
01724  PROG1300
01724  PROG1310
01724  PROG1320
01724  PROG1330
01724  PROG1340
01724  PROG1350
01724  PROG1360
01724  PROG1370
01724  PROG1380
01724  PROG1390
01724  PROG1400
01724  PROG1410
01724  PROG1420
01724  PROG1430
01724  PROG1440
01724  PROG1450
01724  PROG1460
01724  PROG1470
01724  PROG1480
01724  PROG1490
01724  PROG1500
01724  PROG1510
01724  PROG1520

POST PROCESSOR ASSEMBLY DATA

1727 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM.

REFERENCES TO DEFINED SYMBOLS
1724  LTH  2, 1727
2  PROG 1727

NO ERROR IN ABOVE ASSEMBLY.
*TIME SPENT IN FAP ..  000007 IN HUNDREDTHS OF MINUTES.
1.17 MINUTES ELAPSED SINCE START OF JOB

SUBPROGRAM STORAGE MAP FOLLOWS.

<table>
<thead>
<tr>
<th>NAME</th>
<th>ORIGIN</th>
<th>ENTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAP</td>
<td>144</td>
<td>151</td>
</tr>
<tr>
<td>PASS1</td>
<td>201</td>
<td>213</td>
</tr>
<tr>
<td>PASS2</td>
<td>317</td>
<td>327</td>
</tr>
<tr>
<td>VAREVL</td>
<td>637</td>
<td>644</td>
</tr>
<tr>
<td>OPTBL</td>
<td>1117</td>
<td>1121</td>
</tr>
<tr>
<td>INTOP</td>
<td>1226</td>
<td>1230</td>
</tr>
<tr>
<td>SCAN</td>
<td>1351</td>
<td>1353</td>
</tr>
<tr>
<td>SYMSTO</td>
<td>2462</td>
<td>2465</td>
</tr>
<tr>
<td>ENDO1</td>
<td>3036</td>
<td>3042</td>
</tr>
<tr>
<td>COMPOP</td>
<td>3207</td>
<td>3217</td>
</tr>
<tr>
<td>EXPR</td>
<td>3752</td>
<td>3760</td>
</tr>
<tr>
<td>TERM</td>
<td>4155</td>
<td>4163</td>
</tr>
<tr>
<td>(MAIN)</td>
<td>4320</td>
<td>4321</td>
</tr>
<tr>
<td>TESTS</td>
<td>4325</td>
<td>4331</td>
</tr>
</tbody>
</table>

END OF STORAGE MAP.
TEST OF CAP, BEGIN ASSEMBLY.

```
50000  056000050016  CAP  REM THE FOLLOWING ARE ALL LEGAL CAP INSTRUCTIONS.
50001  460000050021  REM PROGRAM TO COUNT BITS IN AC.
50002  053400450020  COUNT  LDQ ZERO ZERO TEST CELLS
50003  076000000001  STQ BITS ..
50004  076000000001  LOOP  LTX THSX COUNT 36 BITS.
50005  002000050013  LBT NO BIT OR.
50006  002000050022  YES  SLW WORD BIT, SAVE AC.
50007  036100050017  CAL BITS AND INCREMENT COUNT.
50008  036100050017  ACL ONE ..
50009  036100050017  SLW BITS ..
50010  036100050017  CAL WORD RESTORE AC.
50011  036100050017  LGT 1 NEXT BIT.
50012  111000000003  NO  TIX LOOP INDEX.
50013  002100000000  CAL BITS GET COUNT.
50014  002100000000  OCTL 002100000000 STOP WITH TRANSFER TO 70000 OCTAL.
50015  002100000000  DCE 36 ADDRESS IS 36.
50016  000000000000  ZERO  OCTL 000000000000 TRUE ZERO.
50017  040000000044  THSX LAS 36 INCREMENT OF ONE.
50018  040000000044  ONE  THSX 36 ADDRESS IS 36.
50019  040000000044  REM CATA.
50020  000000000000  BITS  OCTL STORAGE FOR BIT COUNT.
50021  000000000000  INT 1 TEMPORARY STORAGE FOR AC.
50022  000000000000  WORD  OCTL 8 INSTRUCTION TO Definitions.
50023  000000000000  ILCD 8 ILLEGAL OPCODE.
50024  000000000000  TRA UNDEF UNDEFINED SYMBOL.
50025  000000000000  INT 1,2,3,7,13A3,9 ERROR IN INTOP.
50026  000000000000  WMW AEN ILLEGAL OPCODE AND UNDEFINED SYMBOL.
50027  000000000000  COMP NO = YES + LOOP
50028  000000000000  CLA YES
50029  000000000000  FAD LOOP
50030  000000000000  COMP COMP = A * B * C / D / E / FLAG
50031  000000000000  STO NO
50032  000000000000  STO LOOP
50033  000000000000  COMP COUNT = 1 = LOOP1 2 - START 4
50034  000000000000  CLA LOOP 1,2
50035  000000000000  FSB START 4
50036  000000000000  STO COUNT 1
50037  000000000000  COMP DZ = E+Z*(A+Z*(C+Z*(B+Z*A))) + DZ
50038  000000000000  LDQ Z
50039  000000000000  FMP D
50040  000000000000  STQ TEM
50041  000000000000  CLA D
50042  000000000000  FAD TEM
50043  000000000000  STQ TEM
50044  000000000000  LDQ Z
50045  000000000000  FMP D
50046  000000000000  STQ TEM
```

TSTCAP00
TSTCAP01
TSTCAP02
TSTCAP03
TSTCAP04
TSTCAP05
TSTCAP06
TSTCAP07
TSTCAP08
TSTCAP09
TSTCAP10
TSTCAP11
TSTCAP12
TSTCAP13
TSTCAP14
TSTCAP15
TSTCAP16
TSTCAP17
TSTCAP18
TSTCAP19
TSTCAP20
TSTCAP21
TSTCAP22
TSTCAP23
TSTCAP24
TSTCAP25
TSTCAP26
TSTCAP27
TSTCAP28
TSTCAP29
TSTCAP30
TSTCAP31
TSTCAP32
TSTCAP33
REM TEST OF PROCESSED MODIFICATIONS TO CAP.
TSTCAP34
0 50112 000000000000 MUL INT 0  MULTIPLY DEFINED SYMBOL.
TSTCAP35
0 50113 000000000000 MUL INT 0  MULTIPLY DEFINED SYMBOL.
TSTCAP36
0 50114 0000000047761 E PZE -32+BITS PZE CODE.
TSTCAP37
0 50115 0000000050114 D MEZ NO+65 MZE CODE.
TSTCAP38
0 50116 000200006001 SWL $+1  $ FOR THIS LOCATION
TSTCAP39
0 50117 0000000077777 CLA $Z+1  $ TEST.
TSTCAP40
0 50120 000000000000 MTH G/3  DIVISION IN ADDRESS
TSTCAP41
0 50121 0000000050154 Z PON (YES-COUNT)/3+P DIVISION WITH DIVISION IN ADDRESS ARITHMETIC.
TSTCAP42
0 50122 000000000000 STO (C-FLAG)/2  DIVISION WITH NEGATIVE ANSWER.
TSTCAP43
0 50123 0002000030657 SLW -8+BITS+1+1+8+Z USE OF $ AS A SYMBOL.
TSTCAP44
0 50124 000000000014 TREF (+C)/5+0 $/, AND ILLEGAL CODE
TSTCAP45
0 50125 0000000050115  * REMARK CARD WITH * IN COLUMN 1.
TSTCAP46
0 50126 000000000012 BLNK BLANK OPCODE.
TSTCAP47
0 50127 000000000000 ($=*)/ NOP UNDEF S, O, AND U FLAGS.
TSTCAP48
0 50130 000000000000 G MUL 5 THIS IS HOLLERTH INFO.
TSTCAP49
0 50131 000000000012 CLA# FLAG USE OF FLAGGED INSTRUCTION.
TSTCAP50
0 50132 0000000031043 FLAG STD =1139 LITERAL.
TSTCAP51
0 50133 0000000031043 CLA =1139 SAME LITERAL.
TSTCAP52
0 50134 000000000000  * LAC =5597.0 E, S, AND F FLAGS.
TSTCAP53
0 50135 0000000050121 TXN Z,3.5 TAG AND DECREMENT FIELD.
TSTCAP54
0 50136 000000000000 FAD (YES-COUNT)*(NO-LOOP)*START,2+3*(NO-LOOP)
TSTCAP55
0 50137 000000000021 DZ OCT L17,13,-44,G,13,1 OCT PSEUDO-OP.
TSTCAP56
0 50140 0241000000132 FDG FLAG A DIVIDE INSTRUCTION.
TSTCAP57
0 50141 060010050113 STD MUL=UNDEF FOLLOWED BY A STORE.
TSTCAP58
0 50142 0000000050300 START EQU COUNT+3*(NO-LOOP) PROPER USE OF EQU.
TSTCAP59
0 50143 0002000050142 A TRA START USE OF SYMBOL DEFINED BY EQU.
TSTCAP60
0 50144 000000000126 PI EQU END-CMP PHASE ERROR.
TSTCAP61
0 50145 0002000050144 B TRA PI USE OF SYMMD WITH PHASE ERROR.
TSTCAP62
0 50146 000000000000 E BSS S PROPER BSS.
TSTCAP63
0 50147 000000000000 F BSS LOOP-COUNT PROPER SYMMDIC DEFINITION OF BSS.
TSTCAP64
0 50150 000000000000 LCOPO1 BSS CAL=+ IMPROPER BSS, PHASE ERROR.
TSTCAP65
0 50151 000000000000 CALL AT COUNT-NO,C,YES LEGAL CALL MACRO.
TSTCAP66
0 50152 000000000000 CALL2 CALL ABLE,BAKER+2,CHRLY=5,(NO-LOOP)
TSTCAP67
0 50153 050000000000 FDO CLA S CHECK ILC AFTER BSS AND CALL.
TSTCAP68
0 50000 000000000000 END END COUNT+1=UNDEF FINALLY THE END.
TSTCAP69

RETURN FROM CAP, ENTRY POINT IS 5000CC.
POST MORTEM OF SYMBOLIC PROGRAM.

CAP REM THE FOLLOWING ARE ALL LEGAL CAP INSTRUCTIONS.  TSTCAP00
REM PROGRAM TO COUNT BITS IN AC.  TSTCAP01
COUNT LDQ ZERO ZERO TEST CELLS TSTCAP02
STQ BITS ** TSTCAP03
LOOP LTXH THSX COUNT 36 BITS. TSTCAP04
LBT BIT OR NO. TSTCAP05
TRA NO NO BIT. TSTCAP06
YES SLW WORD BIT, SAVE AC. TSTCAP07
CAL BITS AND INCREMENT COUNT. TSTCAP08
ACL ONE ** TSTCAP09
CAL BITS ** TSTCAP10
CAL WORD RESTORE AC. TSTCAP11
LGR 1 NEXT BIT. TSTCAP12
AQ TIX LCOP INDEX. TSTCAP13
CAL BITS GET COUNT. TSTCAP14
DONE CCTL 00210000700000 STOP WITH TRANSFER TO 70000 OCTAL. TSTCAP15
REM STORAGE. TSTCAP16
ZERO CCTL 00000000000000 TRUE ZERO. TSTCAP17
ONE INT 1 INCREMENT OF ONE. TSTCAP18
THSX LAS 36 ADDRESS IS 36. TSTCAP19
REM DATA. TSTCAP20
BITS INT 0 STORAGE FOR BIT COUNT. TSTCAP21
WORD INT 0 TEMPORARY STORAGE FOR AC. TSTCAP22
REM TEST OF CAP PSEUDO-OPS, AND FLAGS. TSTCAP23
ILDC B ILLEGAL OPCODE. TSTCAP24
TRA UNDEF UNDEFINED SYMBOL. TSTCAP25
INT 13,7,13A3,9 ERROR IN INTP. TSTCAP26
WMW AEN ILLEGAL OPCODE AND UNDEFINED SYMBOL. TSTCAP27
COMP NO = YES + LOOP TSTCAP28
COMP COMP29 = A * E / C / D / E = FLAG TSTCAP29
TRA COMP USE OF SYMBOL DEFINED BY COMP. TSTCAP30
COMP COUNT, 1 = LCOP01Z - START, 4 TSTCAP31
COMP DZ = (E+Z*(C+Z*(B+Z*A))) + FC0 TSTCAP32
TIX COUNT=3*{(NO=YES)+Z+111}+ (LOOP-COUNT)+8*(ZERO-DONE)+8 TSTCAP33
REM TEST OF PROPOSED MODIFICATIONS TO CAP. TSTCAP34
MUL INT 0 MULTIPLY DEFINED SYMBOL. TSTCAP35
MUL INT 0 MULTIPLY DEFINED SYMBOL. TSTCAP36
C PZE -32+BITS PZE CODE. TSTCAP37
D MZE NO=65 MZE CODE. TSTCAP38
SLW $+1 FOR THIS LOCATION TSTCAP39
CLA 2*1 FOR TEST. TSTCAP40
MTH C/3 DIVISION IN ADDRESS TSTCAP41
Z PON (YES-COUNT)+3F PON WITH DIVISION IN ADDRESS ARITHMETIC. TSTCAP42
STD (C-FLAG)Z/2 DIVISION WITH NEGATIVE ANSWER. TSTCAP43
SLW -8+BITS*$+1+7*Z USE OF $ AS A SYMBOL. TSTCAP44
ARF D+$-1/S+Q $,/, AND ILLEGAL CPCODE TSTCAP45
* REMARK Card with * IN COLUMN 1 = TSTCAP46
BLNK BLNK BLANK OPCODE. TSTCAP47
($)= NO UNDEF S,0, AND U FLAGS. TSTCAP48
C MOL 5 THIS IS HOLLERITH INFO. TSTCAP49
CLA* FLAG USE OF FLAGGED INSTRUCTION. TSTCAP50
STD #1139 LITERAL. TSTCAP51
CLA #1139 SAME LITERAL. TSTCAP52
* LAC#5597.0 E, S, AND F FLAGS. TSTCAP53
TNX Z+3.5 TAG AND DECREMENT FIELD. TSTCAP54
FAD (YES-COUNT)+(NO-LOOP)+START, 2+3* (NO-LOOP) TSTCAP55
OCT OCT 17,13,-44,6,13,1 OCT PSEUDO-OP. TSTCAP56
FDP FLAG A DIVIDE INSTRUCTION. TSTCAP57
START EQU COUNT+3*(NO-LOOP) PROPER USE OF EQU.
A TRA START USE OF SYMBOL DEFINED BY EQU.
PI EQU END-COMP PHASE ERROR.
B TRA PI USE OF SYMBOL WITH PHASE ERROR.
E BSS 5 PROPER BSS.
F BSS LOOP-COUNT PROPER SYMBOLIC DEFINITION OF BSS.
LCOP01 BSS CALL-F IMPROPER BSS, PHASE ERROR.
CALL CALL COUNT,NO,C,YES LEGAL CALL MACRO.
CALL2 CALLABLE,BAKER+2,CHRLY-5*(NO-LOOP)
F00 CLA $ CHECK ILC AFTER BSS AND CALL.
END END COUNT+1*UNDEF FINALLY THE ENC.
POST MORTEM OF COLLATION TAPE.

CAP
REM THE FOLLOWING ARE ALL LEGAL CAP INSTRUCTIONS.
CAP
REM PROGRAM TO COUNT BITS IN AC.
COUNT
LDQ ZERO ZERO TEST CELLS
STQ BITS **
LXA THSX COUNT 36 BITS.
LOOP
LBT BIT OR NO.
TRA NO NO BIT.
YES
SLW WORC BIT, SAVE AC,
CAL BITS AND INCREMENT COUNT.
ACL ONE **
SLW BITS **
CAL WORD RESTORE AC.
LGR 1 NEXT BIT.
NO
TIX LOOP INDEX.
CAL BITS GET COUNT.
DONE
GCTL 002100070000 STOP WITH TRANSFER TO 70000 OCTAL.
REM STORAGE.
ZERO
GCTL 000000000000 TRUE ZERO.
ONE
INT 1 INCREMENT OF ONE.
THSX
LAS 36 ADDRESS IS 36.
REM DATA.
BITS
INT 0 STORAGE FOR BIT COUNT.
WORD
INT 0 TEMPORARY STORAGE FOR AC.
REM TEST OF CAP PSEUDO-OPS, AND FLAGS.
ILCD 8 ILLEGAL OPCODE.
TRA UNDEF UNDEFINED SYMBOL.
INT 1, 2, -7, 133, 9 ERROR IN INTOP.
WMW AEN ILLEGAL OPCODE AND UNDEFINED SYMBOL.
COMP
NO = YES + LOOP
CLA YES
FAO LOOP
STO NO
COMP
Z = A • B • C / D / E / FLAG
LDQ A
FMP B
XCA
FMP C
FDP D
XCA
FDP E
XCA
FDP FLAG
STQ TEM
CLA TEM
STO Z
TRA COMP USE OF SYMBOL DEFINED BY COMP.
COMP
COUNT1 = LOOPC1, Z = START+4
CLA
LOOPC1, 2
FSB
START+4
STO
COUNT1
COMP
DG = (E+Z*(D+Z*(C+Z*(B+Z*A)))) + FCO
LDQ Z
FMP A
STO TEM
CLA B
FAO TEM
STO TEM+1
LDQ Z
FMP TEM+1
STO TEM+2
CLA C
FAD TEM+2
STO TEM+3
LDQ Z
FMP TEM+3
STO TEM+4
CLA D
FAD TEM+4
STO TEM+5
LDQ Z
FMP TEM+5
STO TEM+6
CLA E
FAD TEM+6
STO TEM+7
CLA TEM+7
FAD F00
STO D2

TIX COUNT+3*((NO-YES)*Z+1)*((LOOP-COUNT)+8*(ZERO-DONE)+8)
REM TEST OF PROPOSED MODIFICATIONS TO CAP.
STO COUNT+3
MUL INT 0 
MULTIPLY DEFINED SYMBOL.
STO COUNT+4
PUL INT 0 
MULTIPLY DEFINED SYMBOL.
STO COUNT+5
C  PZE -32+BITS
PZE CODE.
STO COUNT+6
D MZE NC+65
MZE CODE.
STO COUNT+7
SLW $+1
$ FOR THIS LOCATION
STO COUNT+8
CLA 2:0+1
$ TEST.
STO COUNT+9
MTY C/3
DIVISION IN ADDRESS
STO COUNT+10
Z PON (YES-COUNT)/3+F
PON WITH DIVISION IN ADDRESS ARITHMETIC.
STO COUNT+11
(C-FLAG)/2
DIVISION WITH NEGATIVE ANSWER.
STO COUNT+12
SLW -8:BITS+$+1+7+2
USE OF $ AS A SYMBOL.
STO COUNT+13
ARF ($-C)/5+G
$/, AND ILLEGAL OPCODE
STO COUNT+14
REM CARD WITH  * IN COLUMN 1.
BLNK BLNK
BLANK OPCODE.
1111 NOP UNDEF
S, O, AND U FLAGS.
W HOL THIS IS HOLLERITH INFO.
CLA* FLAG
USE OF FLAGGED INSTRUCTION.
FLAG STO =1139
LITERAL.
CLA =1139
SAME LITERAL.
* LACX =597.0
E, S, AND F FLAGS.
TXN Z=3.5
TAG AND DECREMENT FIELD.
FAD (YES-COUNT)*(NO-LOOP)+START,2+3*(NO-LOOP)
STO COUNT+15
D2 OCT 17,13,-44,G,13,1
OCT PSEUDO-OP.
FDP DAIN
A DIVIDE INSTRUCTION.
STO COUNT+16
MUL+UNDEF FOLLOWED BY A STORE.
START EQU COUNT+3*(NO-LOOP)
PROPER USE OF EQU.
A TRA START
USE OF SYMBOL DEFINED BY EQU.
PI EQU END-COMP
PHASE ERROR.
B TRA PI
USE OF SYMBOL WITH PHASE ERROR.
E BSS 5
PROPER BSS.
F BSS LOOP-COUNT
PROPER SYMBOLIC DEFINITION OF BSS.
LOOP01 BSS IMPROPER BSS, PHASE ERROR.
CALL CALL COUNT:NO,C+YES LEGAL CALL MACRO.
CALL2 CALL ABLE,BAKER+2,CHRLY-5*(NO-LOOP)
F00 CLA S
CALL2 C:CK ILC AFTER BSS AND CALL.
D2 TEMP TEMPORARY STORAGE AREA BEGINS HERE.
END END COUNT:+1*UNDEF FINALLY THE END.
### POST MORTEM OF CONSOLE

| $s, q, p$ | $0, 0, 0$ |
| AC | $000000050000$ |
| MQ | $6060606060500$ |
| SI | $000000000007$ |
| IR1 | $27624$ |
| IR2 | $1$ |
| IR4 | $73226$ |

### CCTL DUMP OF CAP FOLLOWS

| 144 | TTR 0021000000213 |
| 152 | TTD 0610000000171 |
| 160 | TDI 0410000000200 |
| 174 | TMS 0740000000146 |
| 205 | TTR 00210000007615 |
| 215 | TRA 0200000000220 |
| 223 | TSS 0740000000203 |
| 231 | TDL 4500000000301 |
| 237 | TRA 0200000000247 |
| 245 | TRA 0200000000266 |
| 253 | TRA 0200000000216 |
| 261 | MTR 0000000000301 |
| 267 | TSS 0740000000201 |
| 275 | PAC 0730000000000 |
| 303 | TX 234664456320 |
| 311 | TNX 606060660606 |

### OCTAL DUMP OF PASS1 FOLLOWS

| 144 | TTR 0021000000213 |
| 160 | TDI 0410000000171 |
| 174 | TMS 0740000000146 |
| 215 | TRA 0200000000220 |
| 223 | TSS 0740000000203 |
| 231 | TDL 4500000000301 |
| 237 | TRA 0200000000247 |
| 245 | TRA 0200000000266 |
| 253 | TRA 0200000000216 |
| 261 | MTR 0000000000301 |
| 267 | TSS 0740000000201 |
| 275 | PAC 0730000000000 |
| 303 | TX 234664456320 |
| 311 | TNX 606060660606 |

### CCTL DUMP OF PASS2 FOLLOWS

| 317 | TTR 0021000001211 |
| 333 | STA 0620000000356 |
| 341 | OSI 0440000000602 |
| 347 | CAL 4500000000614 |
| 355 | TX 200002400352 |
| 363 | SIR 005000000002 |
| 371 | HTR 0000000000613 |
| 377 | HTR 0000000001563 |
| 405 | HTR 00000000054526 |
| 413 | TSS 074000000461 |
| 421 | RQL 477300000003 |
| 427 | TX 177777100341 |
| 435 | TSX 074000000523 |
| 443 | ORA 4500100000635 |
| 451 | SLW 0602000000607 |
| 457 | OSI 0440000000602 |
| 465 | PAX 075400000000 |
| 473 | STL 460000000601 |
| 501 | LSG 056000000631 |
| 507 | HTR 000002300006 |

### OCTAL DUMP OF CAP FOLLOWS

| 144 | TTR 0021000000213 |
| 152 | TTD 0610000000171 |
| 160 | TDI 0410000000200 |
| 174 | TMS 0740000000146 |
| 215 | TRA 0200000000220 |
| 223 | TSS 0740000000203 |
| 231 | TDL 4500000000301 |
| 237 | TRA 0200000000247 |
| 245 | TRA 0200000000266 |
| 253 | TRA 0200000000216 |
| 261 | MTR 0000000000301 |
| 267 | TSS 0740000000201 |
| 275 | PAC 0730000000000 |
| 303 | TX 234664456320 |
| 311 | TNX 606060660606 |

### OCTAL DUMP OF PASS1 FOLLOWS

| 144 | TTR 0021000000213 |
| 152 | TTD 0610000000171 |
| 160 | TDI 0410000000200 |
| 174 | TMS 0740000000146 |
| 215 | TRA 0200000000220 |
| 223 | TSS 0740000000203 |
| 231 | TDL 4500000000301 |
| 237 | TRA 0200000000247 |
| 245 | TRA 0200000000266 |
| 253 | TRA 0200000000216 |
| 261 | MTR 0000000000301 |
| 267 | TSS 0740000000201 |
| 275 | PAC 0730000000000 |
| 303 | TX 234664456320 |
| 311 | TNX 606060660606 |

### OCTAL DUMP OF PASS2 FOLLOWS

| 317 | TTR 0021000001211 |
| 333 | STA 0620000000356 |
| 341 | OSI 0440000000602 |
| 347 | CAL 4500000000614 |
| 355 | TX 200002400352 |
| 363 | SIR 005000000002 |
| 371 | HTR 0000000000613 |
| 377 | HTR 0000000001563 |
| 405 | HTR 00000000054526 |
| 413 | TSS 074000000461 |
| 421 | RQL 477300000003 |
| 427 | TX 177777100341 |
| 435 | TSX 074000000523 |
| 443 | ORA 4500100000635 |
| 451 | SLW 0602000000607 |
| 457 | OSI 0440000000602 |
| 465 | PAX 075400000000 |
| 473 | STL 460000000601 |
| 501 | LSG 056000000631 |
| 507 | HTR 000002300006 |
**CCTAL DUMP CF ENDPF FOLLOWS.**

<table>
<thead>
<tr>
<th>3036</th>
<th>TTR 002100002465</th>
</tr>
</thead>
<tbody>
<tr>
<td>3044</td>
<td>SXA 063400403053</td>
</tr>
<tr>
<td>3052</td>
<td>SXI 463400403053</td>
</tr>
<tr>
<td>3060</td>
<td>TIX 632544746511</td>
</tr>
<tr>
<td>3066</td>
<td>TXH 606606606060</td>
</tr>
<tr>
<td>3074</td>
<td>SRA 063400431011</td>
</tr>
<tr>
<td>3104</td>
<td>SRA 063400431016</td>
</tr>
<tr>
<td>3140</td>
<td>SRB 450000032306</td>
</tr>
<tr>
<td>3146</td>
<td>SRA 430000032305</td>
</tr>
<tr>
<td>3154</td>
<td>STD 460000031162</td>
</tr>
<tr>
<td>3170</td>
<td>TNX 606606606060</td>
</tr>
<tr>
<td>3176</td>
<td>TNX 606606606060</td>
</tr>
<tr>
<td>3204</td>
<td>TNX 632544200000</td>
</tr>
</tbody>
</table>

**CCTAL DUMP OF COMPD FOLLOWS.**

<table>
<thead>
<tr>
<th>3207</th>
<th>TTR 002100003111</th>
</tr>
</thead>
<tbody>
<tr>
<td>3215</td>
<td>CAL 450000040001</td>
</tr>
<tr>
<td>3233</td>
<td>SXL 060200003424</td>
</tr>
<tr>
<td>3237</td>
<td>ACT 077400200012</td>
</tr>
<tr>
<td>3245</td>
<td>XCL 413000000000</td>
</tr>
<tr>
<td>3283</td>
<td>TRA 020000003270</td>
</tr>
<tr>
<td>3286</td>
<td>RED 475000000006</td>
</tr>
<tr>
<td>3287</td>
<td>SLH 060600002434</td>
</tr>
<tr>
<td>3275</td>
<td>HTR 000000000020</td>
</tr>
<tr>
<td>3297</td>
<td>LGS 056000003751</td>
</tr>
<tr>
<td>3306</td>
<td>TXN 606606606060</td>
</tr>
</tbody>
</table>

**FOLLOWING 198 CELLS ALL CONTAIN HTR 000000000000**

**CCTAL DUMP CF EXPR FOLLOWS.**

<table>
<thead>
<tr>
<th>3743</th>
<th>HTR 000000000001</th>
</tr>
</thead>
<tbody>
<tr>
<td>3751</td>
<td>TXN 606606606060</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3752</th>
<th>TTR 002100001463</th>
</tr>
</thead>
<tbody>
<tr>
<td>3760</td>
<td>SXA 063400401436</td>
</tr>
<tr>
<td>3766</td>
<td>STB 062500004143</td>
</tr>
<tr>
<td>4002</td>
<td>STZ 060000001444</td>
</tr>
<tr>
<td>4010</td>
<td>TZE 010000004025</td>
</tr>
<tr>
<td>4016</td>
<td>TRA 020000004031</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3764</th>
<th>HTR 000000000001</th>
</tr>
</thead>
<tbody>
<tr>
<td>3768</td>
<td>SXL 063400001444</td>
</tr>
<tr>
<td>3770</td>
<td>STA 062500004037</td>
</tr>
<tr>
<td>4004</td>
<td>STZ 060000001444</td>
</tr>
<tr>
<td>4012</td>
<td>TZE 010000004025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3775</th>
<th>HTR 000000000001</th>
</tr>
</thead>
<tbody>
<tr>
<td>3776</td>
<td>SXL 063400001444</td>
</tr>
<tr>
<td>3777</td>
<td>STA 062500004037</td>
</tr>
<tr>
<td>4005</td>
<td>STZ 060000001444</td>
</tr>
<tr>
<td>4013</td>
<td>TZE 010000004025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3780</th>
<th>HTR 000000000001</th>
</tr>
</thead>
<tbody>
<tr>
<td>3781</td>
<td>SXL 063400001444</td>
</tr>
<tr>
<td>3782</td>
<td>STA 062500004037</td>
</tr>
<tr>
<td>4006</td>
<td>STZ 060000001444</td>
</tr>
<tr>
<td>4014</td>
<td>TZE 010000004025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3783</th>
<th>HTR 000000000001</th>
</tr>
</thead>
<tbody>
<tr>
<td>3784</td>
<td>SXL 063400001444</td>
</tr>
<tr>
<td>3785</td>
<td>STA 062500004037</td>
</tr>
<tr>
<td>4007</td>
<td>STZ 060000001444</td>
</tr>
<tr>
<td>4015</td>
<td>TZE 010000004025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3786</th>
<th>HTR 000000000001</th>
</tr>
</thead>
<tbody>
<tr>
<td>3787</td>
<td>SXL 063400001444</td>
</tr>
<tr>
<td>3788</td>
<td>STA 062500004037</td>
</tr>
<tr>
<td>4008</td>
<td>STZ 060000001444</td>
</tr>
<tr>
<td>4016</td>
<td>TRA 020000004031</td>
</tr>
</tbody>
</table>

**HTR 000000000000**
Appendix C

SUGGESTED ADDITIONS TO CAP

This appendix contains a list of suggested modifications to CAP which a student may attempt to make when using CAP as a laboratory exercise. With each modification is given a "point" value which is an indication of the relative difficulty of modification.

The descriptions of many of these additions make reference to similar facilities in FAP (FORTRAN Assembly Program). Detailed information on the operation of the FAP facilities can be obtained from the FAP reference manual.*

C.1 Symbols

1. Add a test for multiply defined symbols and have CAP indicate with an M every operation involving a multiply defined symbol. 40 points

2. Sort the symbol table after PASS1. Beware, this is a difficult modification. If it fails, nothing else in CAP will work properly.
   a. Interchange sort. 40 points
   b. Radix sort or any sort which takes a time comparable to N log N. 75 points

3. Use an exponential table lookup of the sorted symbol table for SYMGET. 75 points

4. Add the pseudo-op EQU which is to operate as in FAP. Check for phase errors, and indicate with a P. 50 points

5. Add a test to flag the eleven illegal characters in the location field. Indicate with an S. 35 points

C.2 Operation Field

1. Add the three-letter prefix codes to CAP as in FAP (that is, PZE, MZE, PØN, etc., and blank field). 25 points

2. Add the pseudo-op ØCT which accepts octal input in the same format as INT. Errors should be indicated with an E. 50 points

3. Add the pseudo-op BSS as in FAP. Check for phase errors and indicate with a P. 40 points

4. Add the pseudo-op HØL which accepts a card in the format of that in Figure C.1. n is a digit from 1 to 9, or if blank or 0 it is assumed to be 10. 50 points

HØL should then use n words of storage for BCI words as the FAP pseudo-op BCI does.

<table>
<thead>
<tr>
<th>1</th>
<th>678</th>
<th>111213</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>HØL</td>
<td>n</td>
<td>n 6-character words of BCI</td>
</tr>
</tbody>
</table>

Figure C.1. Format for HØL pseudo-operations.

5. Add the pseudo-op CALL as in FAP except that:
   a. No transfer vector is formed.
   b. No error words are generated.

6. Allow for indirect addressing of operators with an asterisk.

7. Use an exponential table lookup for the op-table. (Only 25 points if you did this for the symbol table also.)

8. Improve the REM pseudo-op so that blanks replace the letters REM in the assembly listing.

C.3 Variable Field

1. Modify VAREVL to accept a "/" as a break character for division. Be careful of signs.

2. Modify CAP to consider "$" as a symbol (in the variable field) meaning "this location" as does the "%" in FAP.

3. Add decimal integer literals.

4. Modify CAP to accept a tag field and remove the present tags in ØPTBL.

5. Extend 4 so that CAP will also accept a decrement field and remove the present decrement in ØPTBL.

C.4 Assembly Listing

1. After the assembly listing, print a listing of symbols defined and their values.

2. Bonus for literals: After the symbol table, print a listing of literals.

3. Bonus for multiply defined symbols: Before the assembly listing, print a list of multiply defined symbols and their multiple values.

4. Bonus for symbol table: Form a table of undefined symbols and print after the assembly listing.
5. Print ØCT, INT, and CALL in detail mode.

6. Consider a * in column 1 to indicate a remark as in FAP.

7. Improve the assembly listing by separating the fields of the octal words, that is, CLA 64 should print as follows:

   \[ \begin{array}{c}
   0500 00 0 00100 \\
   \text{While TXL 1,1,1 should print} \\
   -3 00001 1 00001 .
   \end{array} \]

   and INT -32 should print

   \[ \begin{array}{c}
   -000040 00000 \\
   \text{while HØL 1 AB should print (if you have added HØL)} \\
   602122606060
   \end{array} \]

8. Print the nonerror indications A, T, and D where applicable. For example, the letter A means either "an instruction normally written with an address does not have one" or "an instruction normally written without an address has one." Similarly for T (tag) and D (decrement).

9. Add the nonerror indications F and Q. F means "a nonindirectly addressable instruction has an indirect address." Q means "the instruction STØ (instead of the probable STQ) follows a divide instruction."

   N.B.: In connection with these last three suggestions (and others) you may note that all operation codes are completely specified by the first four and the last four octal digits. Thus the middle four may be used in ØPTBL for A, T, D, and F information and for controlling printing of instructions. These middle four digits may be masked out of the opcode before inserting in the assembled program.

C.5 Compiler

1. Add diagnostics to CØMP including
   a. Nonzero reduction level,
   b. Illegal grammar, that is,

   multiple "=" signs

   \[ \begin{array}{c}
   A = C(B) \\
   A = C \div (B) \\
   A = C + (^C)
   \end{array} \]

2. Let column 7 be used for continuation cards in the same way that column 6 is in FØRTRAN, or column 11 is in MAD.

3. Add the operator ** to CØMP in such a way that \( A^{**}B \) would be compiled as

   CLA A
   LDQ B
   TSX EXP3, 4

   and the result from EXP3 is left in the AC. The operator ** should be given proper precedence.
4. Modify the compiler to accept integer and floating point constants and form these into a table, say, LIT+00 to LIT+99 at the end of the program after TEM. To convert an integer of magnitude less than $2^{27}$ to floating point, the following sequence of 7090 instructions will work:

- CLA INT
- $\phi$RA = $\phi$233000000000
- FAD = $\phi$233000000000
- ST$\phi$ FLT

C(AC) = address integer
Put in exponent
Normalize
C(FLT) = floating point equivalent of the integer INT

5. Improve the efficiency of the compiler by reducing the number of combinations

- ST$\phi$ TEM+n
- CLA TEM+n

and replacing the combinations

- STQ TEM+n
- CLA TEM+n

and

- ST$\phi$ TEM+n
- LDQ TEM+n

with

XCA